UNIVERSIDAD DE CHILE
FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS
DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN

# INTERFAZ PARA EL CURSO "LA WEB DE DATOS"

MEMORIA PARA OPTAR AL TÍTULO DE
INGENIERO CIVIL EN COMPUTACIÓN

BASTIÁN HERNAN INOSTROZA GUERRERO

PROFESOR GUÍA:
AIDAN HOGAN

MIEMBROS DE LA COMISIÓN:
CLAUDIO GUTIÉRREZ GALLARDO
JOSÉ URZÚA REINOSO

SANTIAGO DE CHILE
2020

# Resumen

El estándar RDF, en conjunto con los distintos lenguajes y esquemas que lo extienden, permiten a las máquinas pasar de su antiguo rol de mensajero pasivo a uno más protagónico, donde las máquinas mismas procesan e intercambian información en variados documentos y bases de datos, entendiendo y pudiendo interpretar los datos que esta información contiene. Como la web fue inicialmente concebida como una plataforma para compartir documentos legibles por humanos, es necesario que nosotros las personas aprendamos a comunicar información en un formato entendible tanto por máquinas como por otros humanos, en pos de aprovechar el gran potencial combinado de máquinas y personas trabajando en conjunto.

Se hace necesario entonces que las personas aprendan sobre la web de datos, comprendiendo como crear y utilizar los grafos RDF, realizar consultas, razonar sobre estos, y validar los mismos sobre restricciones dadas; son estos los contenidos explorados por los estudiantes del curso "La Web de Datos", en que luego de ver la teoría se encuentran con que las herramientas existentes para aplicar los conocimientos recién adquiridos son diversas, pero que a pesar de la existencia de estándares, no están bien integradas, añadiendo una sobrecarga innecesaria a la hora de explotar las capacidades de la web de datos.

Buscando que los estudiantes tengan una única herramienta que cubra los contenidos vistos en el curso para facilitar el aprendizaje de dichos contenidos, en este trabajo de título se presenta RDF Playground, una aplicación web que permite visualizar grafos RDF, validar su contenido con SHACL y ShEx, consultar y actualizar los datos de un grafo utilizando SPARQL, y razonar sobre estos grafos utilizando los esquemas RDFS y OWL 2 RL, ilustrando los cambios y modificaciones hechos al grafo por el razonamiento deductivo. Este sistema busca ser la herramienta que reemplaza la totalidad de las herramientas utilizadas en el curso, entregando a los estudiantes todas las capacidades necesarias para aplicar los contenidos del curso, interactuando con este mediante cualquier navegador web.

Las evaluaciones realizadas en la edición primavera del año 2020 del curso indican que RDF Playground cumple con su objetivo de integrar las distintas herramientas disponibles en una sola interfaz para facilitar el aprendizaje, siendo valorada como herramienta por estudiantes y el cuerpo docente, aunque existen varias oportunidades de mejora en la usabilidad e intuitividad del sistema.

# Agradecimientos

Quiero agradecer a mis padres Marilyn y Hernán, quienes hace 26 años decidieron embarcarse en la aventura de tener a su primer hijo; ser padre es algo que se va aprendiendo sobre la marcha y no podría estar más agradecido las enormes cantidades de amor y dedicación que han depositado en tamaña empresa; también quiero agradecerles las conversaciones, enseñanzas y reflexiones de sobremesa, y que desde que era pequeño tuvieron oído tanto para mis opiniones más profundas como mis disparates. A mi hermano Gabriel por siempre estar allí, especialmente en las malas, por su compañía y las largas conversaciones sobre los temas más diversos, por saber siempre dar una nueva perspectiva a mis ideas y por todo el cariño y amor entregado durante toda la vida. A mis abuelos por su amor, cuidados, apoyo, "malcrianzas", compañía y enseñanzas. A mis tíos Gabriel y Jorge (que en paz descansa), que sin ser conscientes de ello han marcado mi forma de ver la vida, el mundo que nos rodea, a la sociedad y a mí mismo.

Agradezco a las personas que me han acompañado y aguantado en este largo camino por la Universidad, a la gente de La Radio Integral (que fue mi segunda casa por al menos dos años), a los compañeros del Ciencias de la Computación (que fue mi tercera casa por el resto del tiempo) y del Diario Integral, a los funcionarios y profesores, en particular a Aidan, mi profesor guía por la confianza depositada, del Departamento de Ciencias de la Computación por su dedicación, cariño y atención a lo que hacen.

A Francisca y Eduardo, que me acompañaron desde el primer año de carrera en un 2013 que hoy parece tan distante, por las conversas, la compañía y las aventuras en este largo proceso; ustedes son el corazón de la familia extendida que se ha formado en los años de universidad. Agradezco también a Andrea que me ha acompañado, ayudado y apoyado durante el último tramo de esta gran etapa que ha sido la Universidad.

Gracias a todos por hacer de esta etapa mucho más que clases, controles tareas y proyectos, por hacer que me lleve de esta etapa muy buenos amigos, recuerdos, historias y aventuras. Y finalmente, al Estado de Chile por liberar del peso de la deuda universitaria a mi familia a través de la gratuidad.

# Tabla de Contenido

# Capítulo 1

# Introducción

Una limitación de la web es que fue concebida como una plataforma para compartir documentos solo legibles por otros humanos. Las máquinas por otro lado solían jugar el rol de mensajero pasivo, siendo incapaces de ver la información que transportan. La ambición de la web de datos es evolucionar la web con estándares y prácticas donde máquinas puedan intercambiar información en un formato que puedan procesar con mayor profundidad y exactitud, basado en la idea de que si las máquinas pueden compartir información con más precisión que con las páginas actuales, podrían hacer más cosas en la web, llevando mayor automatización a tareas que las personas debemos realizar manualmente, volviéndolas menos tediosas, más precisas y/o posibles por primera vez.

Entonces dadas las habilidades de las máquinas para procesar información ¿Qué modelo de datos usaremos para estructurar el contenido de la web? Los grafos son más flexibles que las bases de datos relacionales, y son en lo que se basa el estándar RDF de la W3C, que utiliza triples "sujeto-predicado-objeto" como base y dadas sus características permite mejorar el modelo de datos y mezclar información de dos fuentes incluso si los modelos de datos de las fuentes difieren, además de extender la estructura de enlaces de la web para hacer referencia los conceptos [19].

Además, a la hora de utilizar la información la W3C también provee un estándar, el lenguaje declarativo SPARQL, que permite recibir y manipular información en formato RDF; posee una serie de operaciones de consulta analítica incluyendo conjunciones y disyunciones en una sintaxis análoga a SQL clásico [22].

Uno de los beneficios de modelar datos usando grafos es la flexibilidad; esto trae consigo problemas como el asegurarse de que los datos sean consistentes con el esquema y estándar que se aplica. Para validar datos en RDF existen dos estándares: SHACL (*Shapes Constraint Language*) y ShEx (*Shape Expressions*). SHACL valida grafos *de datos* frente a una o varias condiciones expresadas en un grafo *de forma* (en RDF); estas condiciones se consideran una descripción del grafo de datos y se pueden usar también como constructoras de interfaces o como base para integrar datos [12]. ShEx también describe un grafo de datos RDF identificando predicados con sus cardinalidades y tipos asociados [15].

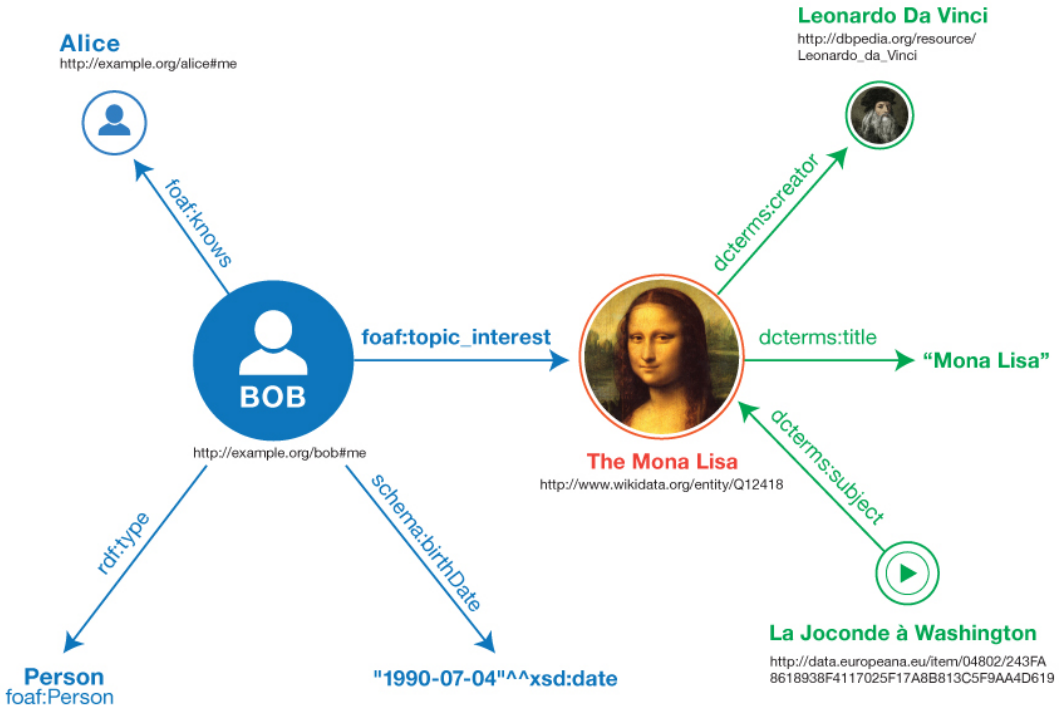



Figura 1.1: Grafo que describe a Bob y su interés en "La Mona Lisa" [21]

Para que las máquinas comprendan la información existen las ontologías, las cuáles engloban una representación, nombre formal y definición de las categorías, propiedades y relaciones entre los conceptos, datos y entidades que establece uno o varios dominios del habla. W3C provee el estándar OWL, diseñado para ser utilizado junto a información descrita en RDF [9].

En el curso "La Web de Datos" los estudiantes aprenden tanto sobre la teoría de la web de datos, así cómo utilizar los estándares más avanzados del estado del arte, aprendiendo a crear, modificar, manipular y consultar grafos RDF en sintaxis *Turtle*, además de validar estos y trabajar con ontologías. El curso incluye laboratorios donde se prueban una o más herramientas por sesión para aprender un tópico determinado de la web de datos; además de un proyecto en el que los alumnos se plantean un desafío y lo enfrentan utilizando las herramientas y conocimientos adquiridos a lo largo del curso.

De forma breve, el problema durante los laboratorios del curso es que se usan varios sistemas disponibles en diferentes sitios web para validar, visualizar, consultar, convertir y razonar sobre los datos, pero muchas veces los sistemas disponibles carecen de características útiles para el curso, y el uso de varios sistemas genera un "hueco" entre los diferentes temas de los laboratorios, ocultando cómo se conectan.

En este trabajo de título se desarrolla RDF Playground, una herramienta computacional de apoyo a la docencia para el curso "La Web de Datos" que permite a los estudiantes tener una experiencia consistente en cuanto a herramientas durante el curso, utilizando la misma interfaz para todos los contenidos y actividades del curso, mostrando en la misma los grafos construidos y manipulados de forma gráfica para facilitar el aprendizaje y manejo de grafos mediante la visualización [6] de estos mismos de una forma clara e interactiva;

además de poder visualizar el resultado de las consultas, verificaciones o razonamientos que se hagan sobre dichos grafos, pudiendo visualizarlos también de forma separada en los casos que corresponda.

## 1.1. Problema Abordado

La principal dificultad a la que se enfrentan aquellos que comienzan a aprender sobre la teoría y aplicación de la web de datos es que las varias herramientas que existen varían mucho en capacidades soportadas, sintaxis de RDF soportadas, tipos de interacción y capacidades básicas de RDF soportadas dentro de las mismas herramientas.

En el caso de las interfaces, en el curso se utilizan: *Easy RDF Converter*, que convierte grafos RDF entre distintas sintaxis incluida *Turtle* y en versiones anteriores permitía exportar el grafo a una imagen de tipo *png* o *svg* (ver sección 2.2.2, se ignora este hecho en el resto de esta sección a fin de explicar su uso en las versiones anteriores del curso); *OWL-RL Reasoner*, que funciona sobre RDFLib y provee razonamiento OWL 2 RL [9] y RDFS [3] a partir de datos en sintaxis *Turtle* o *RDF/XML*; y *Virtuoso*, que permite consultar una base de datos de triples RDF utilizando SPARQL.

Durante el desarrollo del curso los estudiantes utilizan varias de las herramientas anteriormente descritas para objetivos específicos dentro del curso:

1. *EasyRDF* para revisar la correctitud de la sintaxis del grafo descrito y para convertirlo a una imagen.
2. *RDFLib owlrl* en la máquina virtual del curso (mediante *ssh*) para realizar razonamiento con OWL y RDFS.
3. *Virtuoso* en una página expuesta por la máquina virtual del curso para realizar tanto consultas como actualizaciones a una base de triples de datos (*TDB*) con *SPARQL*.

Si bien *SHACL* y *ShEx* son parte del curso, fueron añadidos a los contenidos del curso en primavera de 2019, edición del curso en que no se realizaron laboratorios ni cátedras de estas capacidades por motivos de fuerza mayor, motivo por el cual no se había establecido herramientas para estos dos tópicos al momento de iniciar la presente memoria de título.

*EasyRDF* se ocupa en prácticamente todas las actividades del curso, sola o en conjunto con alguna de las otras herramientas mencionadas; dada su capacidad para ver y verificar la sintaxis de grafos, los estudiantes normalmente escriben sus grafos en esta interfaz, verifican que la sintaxis esté correcta y en ocasiones toman resultados de las otras herramientas para convertirlos en una imagen y visualizarlos.

El constante cambio de herramientas, especialmente cuando se usa al mismo tiempo una que funciona en línea de comandos y otra en web, ralentiza el desarrollo de los laboratorios ya que los estudiantes deben adaptarse a las limitaciones de las herramientas particulares, mover información constantemente entre la máquina virtual y su navegador.

## 1.2. Objetivos

A continuación, se declara el objetivo general y los objetivos específicos de este trabajo de título.

### 1.2.1. Objetivo General

Diseñar y desarrollar una aplicación web que facilite el aprendizaje de los contenidos del curso "La Web de Datos" mediante una interfaz que permita visualizar grafos RDF, y que permita consultar, validar y razonar sobre estos, ilustrando los cambios y modificaciones hechos al grafo en la visualización.

### 1.2.2. Objetivos Específicos

En pos de cumplir el objetivo general planteado se plantean los siguientes objetivos específicos:

1. Entender el estado del arte actual para poder elegir las herramientas más aptas para implementar la interfaz.
2. Entender la carga computacional para poder elegir entre realizar el procesamiento en el servidor o en el cliente.
3. Poder describir un grafo en sintaxis Turtle.
4. Poder visualizar un grafo RDF a partir de un texto escrito en sintaxis Turtle.
5. Poder escribir y ejecutar una consulta SPARQL.
6. Poder razonar sobre el grafo usando RDFS/OWL con retroalimentación visual en la visualización del grafo.
7. Poder validar el grafo usando SHACL/ShEx con retroalimentación visual en la interfaz.
8. Poder visualizar un grafo RDF procesado mostrando de forma distinguible las relaciones originales y las creadas por el procesamiento.
9. Entender cuán bien la interfaz cumple con el objetivo general del trabajo.

## 1.3. Solución Desarrollada

Para cumplir los objetivos planteados se desarrolló una interfaz web que reemplaza todas las herramientas utilizadas actualmente en el curso, sin que estos tengan la necesidad de acceder a una máquina virtual, y permitiendo que puedan visualizar los grafos siempre que sea posible.

La interfaz integra varias de las capacidades y lenguajes utilizados a lo largo del curso, permitiendo al usuario escribir grafos RDF, además de permitir validar, visualizar, consultar,

convertir y razonar sobre estos dentro de la interfaz sin cambiar de herramienta.

Este sistema se compone por un *frontend* y un *backend*, siendo el primero el punto de interacción con los estudiantes, donde estos ingresan la información y reciben los resultados de cualquier procesamiento de dicha información; este es una página estática escrita utilizando el *framework Vue.js* que realiza consultas HTTP a un *backend* para que este procese la información y retorne un resultado; los grafos RDF descritos por el usuario y los recibidos desde el *backend* incluyen siempre que sea posible una visualización de estos, donde los nodos van coloreados por tipo para que el estudiante pueda distinguirlos con facilidad.

Uno de los objetivos del *frontend* es mostrar tanto en el grafo RDF que crea el estudiante, como en los resultados de consultas, razonamiento o validaciones que se expresen en forma de grafo RDF, facilitando la comprensión de los mismos; la visualización se logra utilizando *vis.js*, entregando la capacidad de acercar o alejar la visualización a una parte específica del grafo visualizado, así como mover nodos del grafo según se desee.

El *backend* del sistema se encarga de procesar la información recibida, respondiendo mediante el mismo protocolo (HTTP) el resultado de dicho procesamiento, o bien un mensaje de error; las distintas consultas que acepta el *backend* así como la forma de las respuestas están definidas en una API, donde la recepción de consultas y la entrega de respuestas es manejada utilizando *Spring Boot*.

# Capítulo 2

# Marco Teórico

Las distintas herramientas disponibles en este momento tanto para estudiantes como para especialistas en el área de RDF en general cumplen con una parte de la totalidad de las capacidades existentes en la especificación de RDF, sin importar si son herramientas web o de línea de comandos; en este sentido hay algunas herramientas que permiten explotar más de una capacidad de la especificación a la vez, como por ejemplo *Apache Jena* [26], la que permite: procesar un documento que contenga un grafo en RDF, definirlo, realizar razonamiento sobre este, realizar consultas SPARQL y operar una base de triples de datos.

Para apoyar la realización del curso, es necesario integrar las herramientas existentes de forma coherente, en pos de que el estudiante no se encuentre con una capacidad contenida en el curso que no pueda aplicar en la interfaz, y buscando hacer su uso intuitivo, facilitando el uso de las actividades más recurrentes y manteniendo la "descubribilidad" de las capacidades, incluso para un estudiante que desconozca la totalidad de los contenidos del curso; además buscamos la una manera simple de visualizar los datos, de forma que facilite el análisis de los grafos que se estén trabajando.

## 2.1. Descripción de RDF

En la presente sección se visitan distintos conceptos relacionados al modelo RDF que trataremos a lo largo de este trabajo de título, en pos de explicar el contexto del trabajo de título y los lenguajes y capacidades utilizadas en el mismo.

### 2.1.1. *Resource Description Framework (RDF)*

El *framework* de descripción de recursos (RDF por sus iniciales en inglés) provee el corazón y la base de un modelo de datos para la web de datos (o web semántica). El elemento fundamental del modelo de datos de RDF es el **término RDF**, que es usado para referirse a cualquier *recurso*: cualquier cosa con identidad. Los términos RDF pueden ser divididos en tres subgrupos: *URIs* (o *IRIs*), *literales* y *nodos blancos*.

#### IRIs

El identificador de recursos internacionalizado (o URI, identificador de recursos uniforme) sirve como un identificador global para identificar cualquier recurso, por ejemplo, `https://www.wikidata.org/wiki/Q232141` es utilizado para identificar a la Universidad de Chile en Wikidata [24] (un base de datos RDF creada para tener una fuente común de datos para los distintos proyectos de Wikimedia). En la sintaxis *Turtle* estas son delimitadas con cuñas `<https://es.wikipedia.org/wiki/Wikidata>`; estas pueden ser abreviadas utilizando `prefijo:nombre_local` (por ejemplo `wd:Q232141` para la Universidad de Chile), donde el nombre local es resuelto en el alcance del prefijo definido para generar el IRI completo.

#### Literales

Son una serie de valores léxicos denotados con comillas altas '' en sintaxis *Turtle*; pueden ser tanto planos como tipados.

En el caso de los literales planos se trata de una serie de caracteres encerrados en comillas altas, que pueden incluir un marcador de idioma, como por ejemplo `''Hola Mundo!''@es`.

Los literales tipados están compuestos por un valor léxico y un tipo de datos, como ''2''^^xsd:int; los tipos de datos están identificados por IRIs (como `xsd:int`); *Turtle* provee atajos para los tipos más comunes, como números y booleanos, que sin comillas altas indican su tipo de dato correspondiente.

#### Nodos blancos

Estos son definidos como variables existenciales usadas para denotar la existencia de un recurso sin tener que dar referencia explicita a este usando IRIs o literales. En la práctica sirven como identificadores de alcance local para recursos que no son nombrados; a estos nodos no se les puede hacer referencia fuera de su alcance de origen (un documento RDF). En la sintaxis *Turtle*, los nodos blancos se notan explícitamente con un guión bajo _ a modo de prefijo `_:nodoblanco1`.

### Triples RDF y grafos

Con los términos de RDF establecidos podemos hablar de *triples RDF*, que son utilizados para hacer declaraciones sobre esas cosas; el primer elemento del triple es llamado un *sujeto*, el segundo un *predicado* y el tercero un *objeto*. Un triple RDF puede entonces ser visto como una representación atómica de un hecho o declaración (por ejemplo, `ex:GreenGoblin rel:enemyOf ex:Spiderman` afirma que el sujeto `ex:GreenGoblin` es enemigo de el objeto, en este caso otro IRI, `ex:Spiderman`), lo cual nos sirve para entender el concepto de grafo, el cual se define como una agrupación de declaraciones en la forma de *triples RDF* (ver ejemplo en la figura 2.1, o de forma visual en la figura 2.2).

```
ex:GreenGoblin
 rel:enemyOf ex:Spiderman ;
 a ex:Villain ; # in the context of the Marvel universe
 foaf:name "Green Goblin" .

ex:Spiderman
 a ex:SuperHero ;
 foaf:name "Spiderman" ;
 foaf:knows _:MaryJane .

_:MaryJane foaf:name "Mary Jane" ;
 a foaf:Person .
```

Figura 2.1: Ejemplo de grafo RDF sobre la relación entre *Green Goblin* y *Spiderman*.

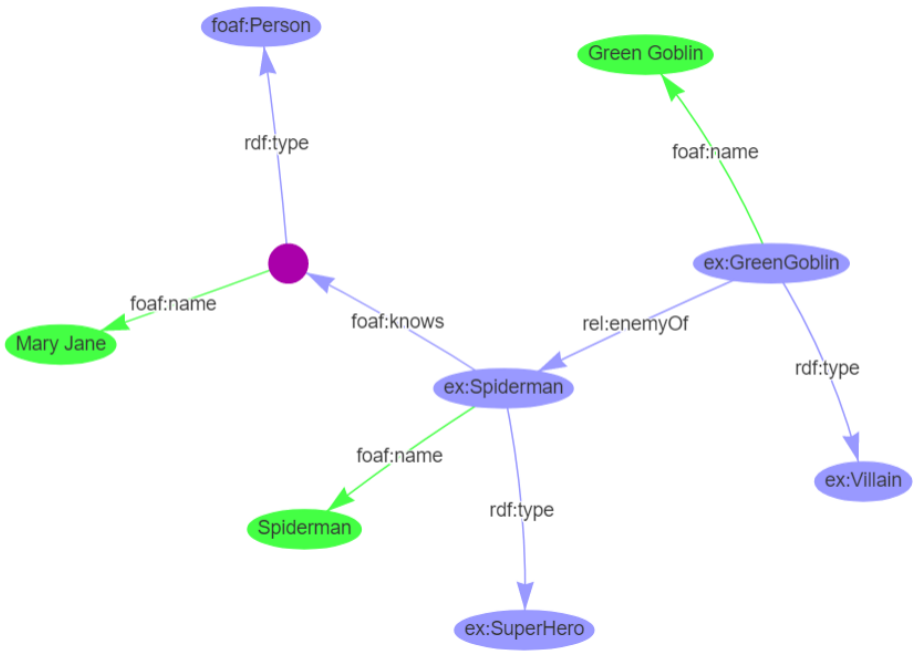



Figura 2.2: Visualización del grafo de la figura 2.1

### 2.1.2. Esquema RDF (RDFS)

Este esquema busca extender el vocabulario para permitir semánticas añadidas a clases y propiedades definidos por el usuario. Dentro de la extensión que hace RDFS a RDF encontramos cuatro términos clave, que se utilizan para definir relaciones entre clases y propiedades: `rdfs:subClassOf`, `rdfs:subPropertyOf`, `rdfs:domain` y `rdfs:range`.

Utilizando el ejemplo en la figura 2.1, podemos expresar que tanto un villano como un superhéroe son personas en el contexto del grafo con las sentencias de la figura 2.3.

```
ex:Villain rdfs:subClassOf foaf:Person .
ex:SuperHero rdfs:subClassOf foaf:Person .
```

Figura 2.3: Ejemplo de uso de RDFS para extender clases definidas en la figura 2.1

### 2.1.3. *Web Ontology Language* (OWL)

Para que las máquinas comprendan la información existen las ontologías, las cuáles engloban una representación, nombre formal y definición de las categorías, propiedades y relaciones entre los conceptos, datos y entidades que establece uno o varios dominios del habla. W3C provee el estándar OWL (Lenguaje de ontología web), diseñado para ser utilizado junto a información descrita en RDF [9]. Este lenguaje pretende extender RDFS con una semántica más expresiva.

Un ejemplo de como podemos utilizar OWL sería el caso de la figura 2.4, en el cual se expresa que la propiedad `rel:enemyOf` es simétrica.

```
rel:enemyOf rdf:type owl:SymmetricProperty .
```

Figura 2.4: Ejemplo de uso de OWL para expresar una característica de la propiedad descrita en la figura 2.1

### 2.1.4. Razonar utilizando RDFS y OWL

Al igual que RDFS puede ser utilizado para obtener de forma deductiva más declaraciones en lo que llamaremos *razonamiento* (también se le llama inferencia) durante este trabajo de título. Durante el curso se trabaja con OWL 2 RL, un perfil de OWL diseñado para ser soportado particularmente por motores de razonamiento, se trata de un subgrupo del perfil OWL 2 DL y es acompañado de varias reglas de enlace con RDF.

Un motor de razonamiento RDFS podría, por ejemplo, con las sentencias de la figura 2.3 inferir para el grafo de la figura 2.1 que tanto `ex:Spiderman` como `ex:GreenGoblin` son personas. En el caso de OWL, la sentencia de la figura 2.4 permitiría a un motor de razonamiento OWL 2 RL, que opere sobre el grafo de la figura 2.1, a inferir que `ex:Spiderman` también es enemigo de `ex:GreenGoblin`, a partir de que `ex:GreenGoblin` es enemigo de `ex:Spiderman`, y la propiedad `rel:enemyOf` es simétrica.

### 2.1.5. SPARQL

El estándar SPARQL es un lenguaje de consultas creado específicamente para los datos RDF [16]. SPARQL es ortogonal a RDFS y OWL, y está construido sobre el modelo de datos de RDF sin soporte para el razonamiento. Es parecido a SQL, compartiendo algunas palabras clave con este. La sintaxis RDF específica de SPARQL es muy cercana a la de *Turtle*, por lo que es fácil para un conocedor de esta entender SPARQL.

Un ejemplo simple sería para el grafo descrito en la figura 2.1 querer obtener todos los nombres (`foaf:name`) de las entidades presentes, lo que se podría realizar con la consulta de la figura 2.5, recibiendo los datos de la figura 2.6.

```
SELECT * WHERE {?s foaf:name ?o}
```

Figura 2.5: Consulta SPARQL para obtener los nombres presentes en el grafo consultado

```
?s ?o
_:MaryJane    "Mary Jane"
ex:Spiderman    "Spiderman"
ex:GreenGoblin    "Green Goblin"
```

Figura 2.6: Respuesta en "TSV" para la consulta de la figura 2.5 sobre el grafo de la figura 2.1

### 2.1.6. SPARQL Update

La especificación original de SPARQL no incluía métodos estandarizados con los cuales el contenido de una base de datos SPARQL pudiese ser actualizada. El estándar SPARQL 1.1 Update [14] intenta rectificar esta situación proveyendo un lenguaje de actualizaciones similar al de consultas.

Por ejemplo si queremos poner al día a nuestro contexto de *Spiderman*, habiendo ingresado el grafo de la figura 2.1 a la base de datos, podríamos añadir a *Iron Man* a este con la instrucción de la figura 2.7.

```
INSERT DATA {
 ex:IronMan
  a ex:SuperHero ;
  foaf:name "Iron Man" ;
  foaf:knows ex:Spiderman .
}
```

Figura 2.7: Actualización de SPARQL Update que añade a *Iron Man* a la base de datos

### 2.1.7. SHACL

El lenguaje de restricción de formas (*Shapes Constraint Language* (SHACL)) es un lenguaje para validar grafos RDF contra un grupo de condiciones [12], dichas condiciones se proveen como formas y otros constructos expresados como un grafo RDF; los grafos que se emplean para expresar las condiciones de esta forma son llamados grafos de forma, mientras que los grafos que son validados contra estas formas son llamados grafos de datos [12].

Si quisiéramos verificar que todos los villanos tengan a lo más un enemigo, podríamos crear la forma de la figura 2.8 para hacerlo utilizando SHACL.

```
ex:VillainShape
 a sh:NodeShape ;
 sh:targetClass ex:Villain ;    # Aplica a todos los Villanos
 sh:property [                  # _:blankNode
     sh:path rel:enemyOf ;  # restringe los valores de rel:enemyOf
     sh:maxCount 1 ;        # tiene un solo enemigo
     sh:nodeKind sh:IRI ;   # el objeto tiene que ser un IRI
 ] .
```

Figura 2.8: Forma en SHACL para validar que los villanos tengan a lo más un enemigo

### 2.1.8. ShEx

El lenguaje de expresiones de forma (o ShEx, abreviación de *Shape Expressions*) describe nodos y estructuras de grafos; una restricción de nodos describe a un nodo RDF y una forma describe triples incluyendo nodos en un grafo RDF; estas descripciones identifican predicados, sus cardinalidades asociadas y tipos. Las formas de ShEx pueden ser utilizadas para validar datos, lo que se realiza a lo largo del curso [15]. ShEx y SHACL poseen varias diferencias y comparten bastantes similitudes, las cuales no ahondaremos en este documento.

Para validar por ejemplo que *Spiderman* cumpla con que su nombre sea de tipo *literal*, permitirle tener cero o más enemigos, y restringir sus conocidos a IRIs y nodos blancos, usamos ShEx como en la figura 2.9 para determinar si `ex:Spiderman` cumple con las condiciones establecidas, en `ex:SuperHeroShape`.

```
ex:SuperHeroShape {
    foaf:name       Literal+;
    rel:enemyOf     IRI*;
    foaf:knows      NonLiteral*
}
```

Figura 2.9: Forma en ShEx que establece condiciones sobre los triples para los superhéroes del grafo 2.1

## 2.2. Revisión de Literatura

Las herramientas disponibles en la actualidad que cumplen con el estándar RDF y permiten a sus usuarios explotar las distintas capacidades del estándar se pueden dividir en dos grupos: interfaces y bibliotecas. A continuación se hace una revisión de las herramientas existentes, revisando sus capacidades, limitaciones y el lenguaje en el que están implementados.

### 2.2.1. Bibliotecas

#### *Apache Jena*

*Apache Jena* (comúnmente mencionado como *Jena*) es un *framework* escrito en Java que apunta a la creación de aplicaciones para la web de datos, y funciona como una colección de APIs que trabajan en conjunto para procesar los datos [23].

Este *framework* cubre las siguientes capacidades importantes para el curso: procesar un documento que contenga un grafo en RDF en varias sintaxis o definirlo en Java mismo con una serie de clases dispuestas por la biblioteca; realizar razonamiento sobre este (sin embargo este razonamiento se limita a RDFS y OWL en su primera versión, en el curso se utiliza RDFS y OW 2 RL); realizar validaciones a un grafo RDF utilizando restricciones definidas en SHACL y realizar consultas SPARQL sobre un grafo cargado a la biblioteca y operar una base de triples de datos, utilizando SPARQL tanto para manipular la base de datos con SPARQL Update, o realizar consultas directamente a la base de datos.

Si bien no permite realizar validaciones con ShEx, la biblioteca *ShExJava* [11] destaca que es capaz de trabajar con la API de modelos RDF de *Apache Jena* para realizar las validaciones.

#### *ShExJava*

Esta biblioteca creada por Inria y escrita en Java provee de validación de expresiones en grafos RDF, utilizando una implementación de ShEx en su versión 2.0 [15].

#### *Eclipse rdf4j*

El *framework rdf4j* funciona en Java y provee una API para ser conectado a alguna base de datos de RDF [7]; al igual que *Apache Jena* se enfoca al desarrollo de aplicaciones de la web de datos, y permite conectarse a otros *endpoints* con bases de triples de datos utilizando SPARQL. Al igual que *Jena* (sección 2.2.1) depende de *ShExJava* para realizar validaciones ShEx, y se debe usar *org.shacl* para las validaciones en SHACL.

#### *RDFLib*

*RDFLib* es un paquete de Python que puede leer, crear y extender grafos en RDF y RDF *generalizado* (soporta utilizar etiquetas además de IRIs en el sujeto de un triple de datos); posee una interfaz que puede ser respaldada por una implementación de almacenamiento [1]; en particular soporta conectarse a *Berkeley DB* logrando con este almacenamiento persistente.

Los creadores de esta herramienta han creado algunas otras bibliotecas que soportan capacidades adicionales, que dependen de *RDFLib* como base, como por ejemplo el validador *pySHACL*, y el razonador *OWL-RL* que implementa un razonamiento para RDFS y OWL 2 RL.

* La biblioteca *PyShEx* creada por Harold Solbrig añade validación ShEx a RDFLib.

#### *rdflib.js*

Esta es una biblioteca escrita en Javascript utilizando *node.js*; dentro de sus características más importantes destacan que posee una API para conectarse a una base de datos, y que posee un subconjunto de consultas de SPARQL utilizables [20]. La documentación del proyecto no está completa, soporta SHACL pero no soporta ShEx.

#### *librdf.org*

Es un grupo de bibliotecas que proveen soporte para RDF escrito en C, con adaptaciones para PHP, Python y Ruby [2]. Entre las capacidades relevantes para el curso se encuentran que puede utilizarse en conjunto a *Virtuoso*, soporta *Turtle* y soporta consultas SPARQL.

### Comparativa

Para sintetizar la discusión previa, en la tabla 2.1, se presenta un resumen de sus capacidades en el contexto de los objetivos de este trabajo, en la cual se puede apreciar las diferencias en capacidades cubiertas por cada herramienta; se denota con el símbolo $*$ cada vez que el soporte posea excepciones, o posea una dependencia no incluida.

| biblioteca | RDF | RDFS | OWL | SPARQL | SHACL | ShEx | Triple DB |
|---|---|---|---|---|---|---|---|
| Apache Jena | ✓ | ✓ | * | ✓ | * | * | ✓ |
| rdf4j | ✓ | ✓ | ✓ | ✓ | * | * | * |
| RDFLib | ✓ | ✓ | X | ✓ | * | * | * |
| rdflib.js | ✓ | ✓ | X | * | ✓ | X | X |
| librdf.org | ✓ | ✓ | X | ✓ | X | X | ✓ |

Tabla 2.1: Tabla comparativa de las capacidades de varias bibliotecas.

### 2.2.2. Interfaces

Actualmente existen algunas interfaces para trabajar con grafos RDF, todas manipulan el grafo sólo en forma de texto. Según nuestros antecedentes no existe una herramienta abierta que combine estos elementos de la web de datos en una interfaz visual, o que integre una buena parte de las capacidades del estándar RDF mencionadas en este texto. A continuación, se mencionan las interfaces utilizadas actualmente en el curso y sus características más relevantes para este trabajo de título.

#### *EasyRDF converter*

Esta página desarrollada en PHP y de código abierto convierte grafos RDF entre varias sintaxis incluida *Turtle*, soportando además DOT (o *Graphviz*) como formato de salida, un lenguaje de descripción de grafos que en general se utiliza para crear visualizaciones de estos. Una de sus características más útiles en el contexto del curso es que revisa la sintaxis del grafo RDF ingresado, mostrando al usuario errores expresivos que permiten identificar fácilmente el problema.

Esta página a la fecha de redacción de este trabajo de título ya no soporta la conversión de grafos a imágenes, permitiendo en cambio la conversión a un texto *Graphviz* (también llamado *DOT*), el cual se puede convertir en una representación visual utilizando una herramienta de terceros (como por ejemplo `graphviz.org`). La página para la redacción de la introducción a la memoria de título, y versiones anteriores del curso, soportaba convertir los grafos RDF descritos a los formatos de imagen `png`, `gif` y `svg` uniendo los prefijos definidos con cada elemento, aumentando considerablemente el tamaño de los nodos.

#### *OWL RL Reasoner*

Este servicio web funciona sobre *RDFLib/OWL-RL* (sección 2.2.1) y provee razonamiento OWL 2 RL [9] a partir de datos en sintaxis *Turtle* o *RDF/XML*; permite utilizar OWL 2 RL y RDFS [3] para inferir e interpretar información sobre un grafo RDF de datos, entregando al usuario del servicio otro grafo RDF que incluye tanto los datos originales como los que el servicio fue capaz de inferir a partir de estos. En el curso "La Web de Datos" existe un capítulo entero dedicado a las ontologías y razonamiento, y se utiliza esta herramienta en los laboratorios.

#### *Virtuoso*

*Virtuoso* es una base de datos que entre otras cosas almacena triples RDF, permite hacer consultas y actualizaciones sobre la información almacenada utilizando SPARQL [25]; entrega por defecto un *endpoint* en la forma de interfaz web al que los estudiantes acceden en el curso, almacenando triples y consultando sobre estos.

# Capítulo 3

# Concepción de la Solución

Con el fin de facilitar el aprendizaje de los contenidos del curso "La Web de Datos", en esta sección presentamos RDF Playground, una interfaz web que ayuda al aprendizaje de los contenidos del curso permitiendo en la misma: visualizar grafos RDF, consultar sobre estos, razonar y validarlos; los grafos RDF visualizados pueden ser los descritos por el estudiante o los que resulten de la interacción con las herramientas que la interfaz provee.

## 3.1. Principales Requisitos de la Solución

En esta sección se describen los principales requisitos de usuario (estudiante) que guiaron el desarrollo de la solución presentada en esta memoria.

### 3.1.1. Revisión la sintaxis de un grafo descrito en *Turtle*

Lo primero se aprende a realizar en el curso es describir grafos en el modelo de RDF utilizado principalmente en el curso, y es de las cosas que más se realizan durante el curso dado a que es el punto de partida para utilizar herramientas que trabajan con dichos grafos. RDF Playground debe permitir al estudiante ingresar una descripción de grafo RDF y revisar la sintaxis de esta en cualquier punto del proceso de descripción; esta revisión debe confirmar claramente cuando la sintaxis de grafo está correcta, junto con una visualización de este; y en caso de que la sintaxis no sea correcta debe indicar con claridad el primer error encontrado en la descripción de esta.

### 3.1.2. Consultas o actualizaciones SPARQL a un grafo dado, o al contenido en la base de triples de datos

El sistema debe permitir al usuario utilizar el lenguaje SPARQL para realizar consultas y/o actualizaciones según corresponda a uno de los siguientes grafos: el contenido en la base de triples de datos del sistema o uno definido por el usuario.

Siempre que el formato de salida del resultado de una consulta o actualización SPARQL sea un lenguaje para describir grafos RDF, el sistema además proveerá una visualización de dicho resultado.

### 3.1.3. Razonar sobre un grafo RDF dado

El sistema debe permitir al estudiante realizar razonamiento con los esquemas RDFS y OWL 2 RL sobre un grafo RDF descrito por este, mostrando el resultado del razonamiento en texto en sintaxis *Turtle*, además de una visualización en la que se distinga con una coloración distinta los nodos y arcos descritos por el estudiante y los agregados fruto del razonador utilizado.

### 3.1.4. Validar que un grafo RDF cumple con las restricciones dadas

El sistema debe validar, conforme a las restricciones descritas por el usuario utilizando el lenguaje SHACL, un grafo RDF entregado por el usuario, reportando el resultado de la validación en texto.

### 3.1.5. Validar que un grafo RDF cumple con una forma dada

El sistema debe validar utilizando el esquema ShEx que ciertos elementos del grafo RDF descrito por el estudiante , utilizando el lenguaje ShEx, cumpla con las restricciones de forma descritas por el estudiante, de acuerdo a una descripción entregada por este de cuáles elementos deben cumplir con cada condición, reportando el resultado de la validación para cada par *elemento–restricción de forma*, mostrando claramente si el resultado de la validación es positivo o negativo.

## 3.2. Historias de Usuario

En la presente sección se describen las principales interacciones a soportar del sistema, las cuales se realizan principalmente entre un estudiante y RDF Playground, utilizando para ello un navegador web; se debe considerar que dado que el uso de la herramienta se da en principio en los laboratorios del curso, sin estar restringido su uso solo a estos, el estudiante

puede solicitar la ayuda del cuerpo docente para utilizar el sistema o resolver dudas propias de los temas tratados en dicho laboratorio. Para ilustrar los conceptos, se usarán capturas de pantalla del sistema final cuyo diseño e implementación serán descritos en los capítulos posteriores.

### 3.2.1. Revisar la validez de la sintaxis utilizada en una descripción de grafo en RDF

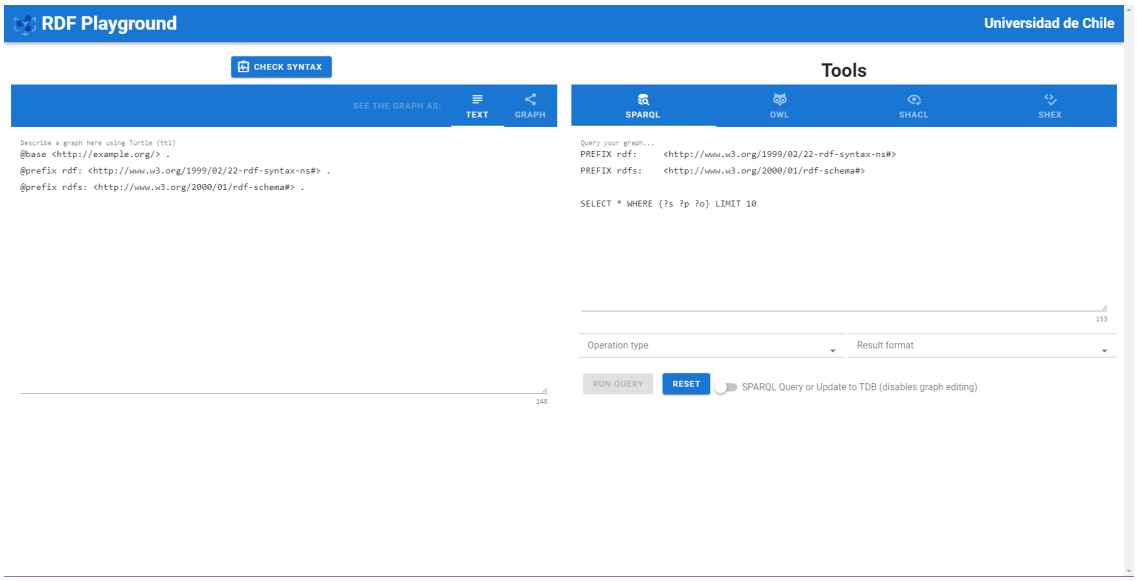
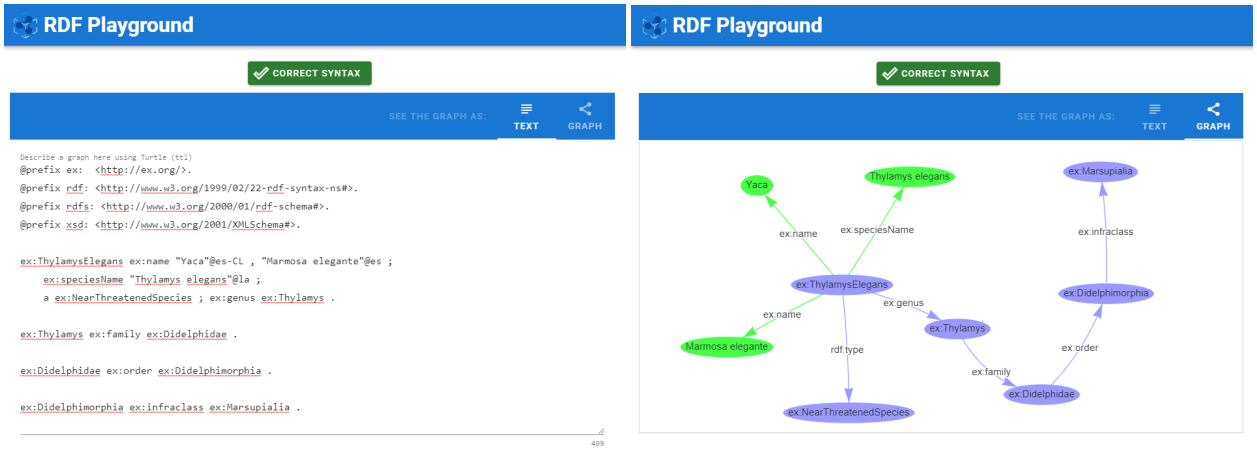



Figura 3.1: Interfaz presentada al usuario al entrar a la página

El estudiante entra a la página de la herramienta, RDF Playground, que utilizarán durante el laboratorio, y se encuentra con la vista principal de la interfaz, ilustrada en la figura 3.1, en la cual se le indica que debe describir un grafo RDF a partir de un texto en prosa dado. El estudiante ingresa en el campo de texto de la mitad izquierda (figura 3.1) su descripción en RDF para el texto dado, para luego presionar el botón "*Check Syntax*" que se encuentra sobre el campo de texto. RDF Playground luego retorna el resultado de la revisión de sintaxis, que en este caso muestra que la sintaxis es apropiada (figura 3.2a), habilitando la visualización del grafo, a la cual el estudiante accede presionando en la pestaña "*Graph*" del lado izquierdo (figura 3.2b).

### 3.2.2. Realizar consulta en SPARQL sobre un grafo descrito

Para este caso de uso se considera que el estudiante ya ha descrito un grafo RDF correctamente utilizando la sintaxis turtle (caso de uso 3.2.1); luego de lo cual el estudiante debe para el laboratorio consultar sobre este grafo (en este caso se realiza una consulta por una selección de todos los nombres `[ex:name]` que aparezcan en el grafo). Para eso el estudiante rellena su consulta en la mitad derecha de la interfaz (zona de herramientas) en la pestaña de SPARQL; luego marca en el menú desplegable de la izquierda, correspondiente al tipo de

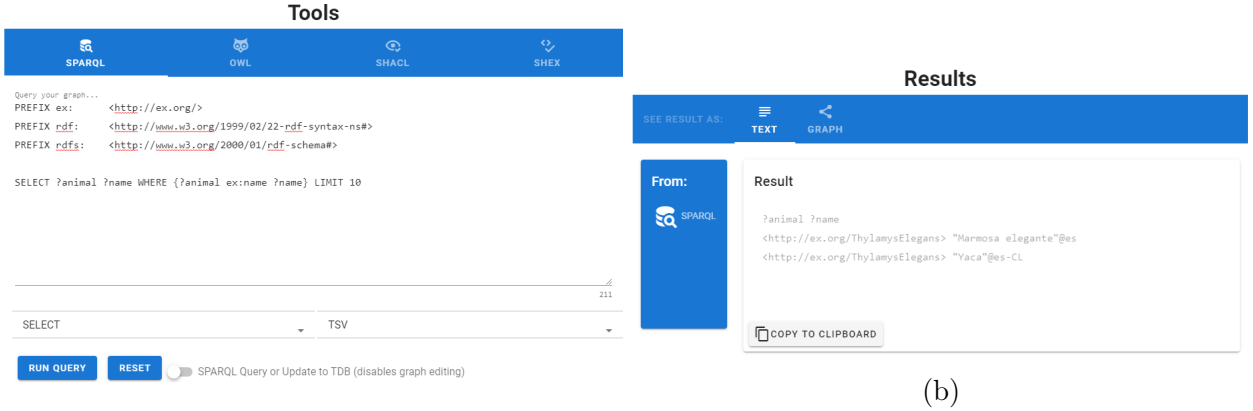



(a) (b)

Figura 3.2: (a) descripción de un grafo bien descrito en sintaxis *Turtle*. (b) visualización del mismo grafo correctamente descrito

consulta, la opción "*SELECT*", y en el menú desplegable de la derecha, correspondiente al formato del resultado, la opción "*TSV*".

Finalmente presiona el botón "*RUN QUERY*", y RDF Playground despliega una sección de resultados, en la que se ve a la izquierda la herramienta que originó el resultado (SPARQL) y el texto resultante de la consulta.



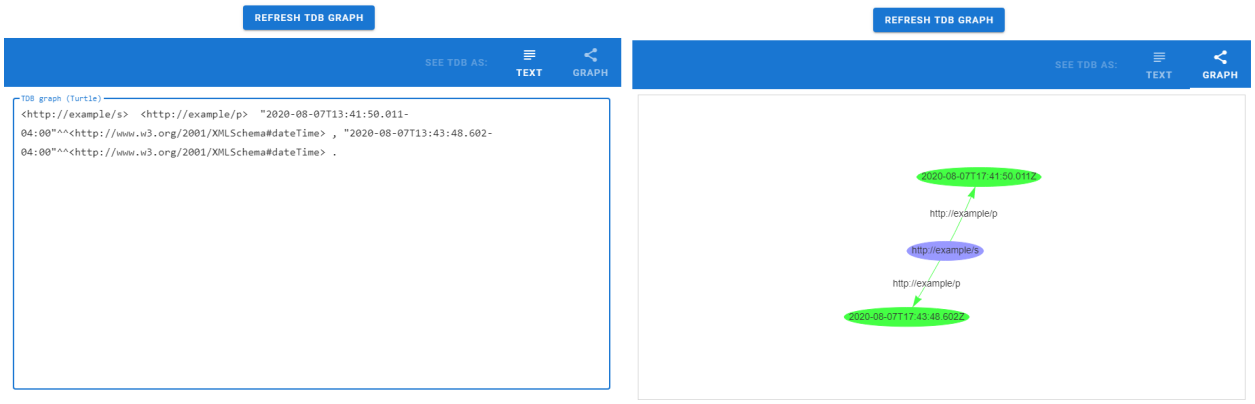

(a) (b)

Figura 3.3: (a) área de la interfaz para describir una consulta en SPARQL. (b) respuesta de la interfaz a la consulta en la sección de resultados

### 3.2.3. Realizar una SPARQL Update a la base de triples de datos

En esta ocasión el estudiante debe guardar en la base de triples de datos (TDB por su sigla en inglés) que expone la interfaz un grafo anteriormente descrito (en particular el de la historia de usuario 3.2.1); para esto en la pestaña de SPARQL de la zona de herramientas (visible en la figura 3.3a) el estudiante habilita la opción "SPARQL *Query or Update to TDB (Disables graph editing)*".

La opción de hacer consultas o actualizaciones a la base de triples de datos modifica el lado de la izquierda de la interfaz, el que ahora muestra el contenido de la base de triples de datos (figura 3.4a), además de permitir visualizar este (figura 3.4b); además cambia el botón "*RUN QUERY*" por el botón "*QUERY TDB*".




(a) (b)

Figura 3.4: (a) contenido de la TDB en *Turtle* antes de realizar la actualización. (b) visualización del contenido de la TDB.

Luego el estudiante completa su actualización de SPARQL en el área de texto de la pestaña SPARQL de la zona de herramientas (ver figura 3.5a), y presiona el botón "*QUERY TDB*"; RDF Playground procesa la actualización y responde en el área de resultados un mensaje con el resultado de la operación, dejando en claro que fue correctamente realizada (ver figura 3.5b). Para ver la base de triples de datos con la nueva información el estudiante presiona el botón "*REFRESH TDB GRAPH*", lo cual hace que se muestre la información actualizada de la base de triples de datos (figura 3.5c), y en su visualización respectiva (figura 3.5d).

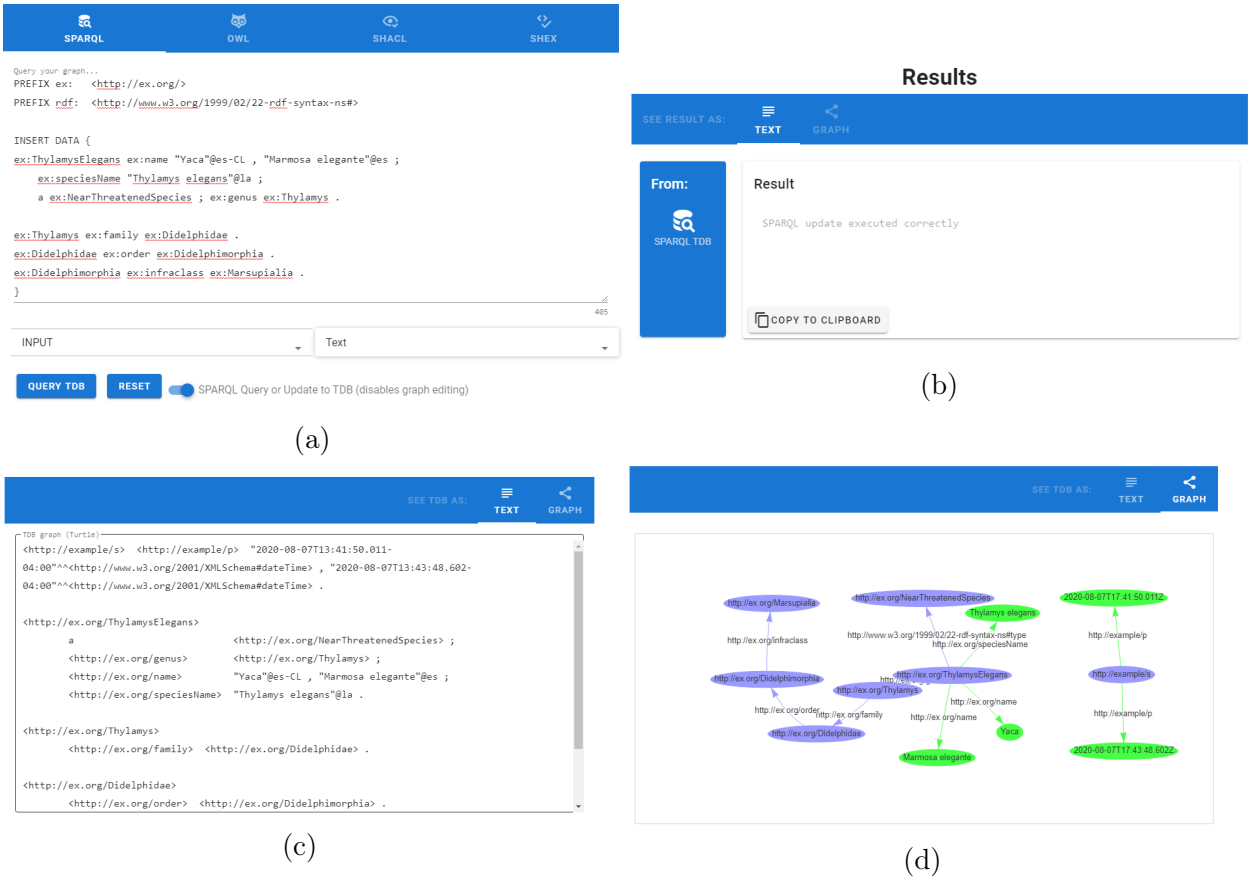


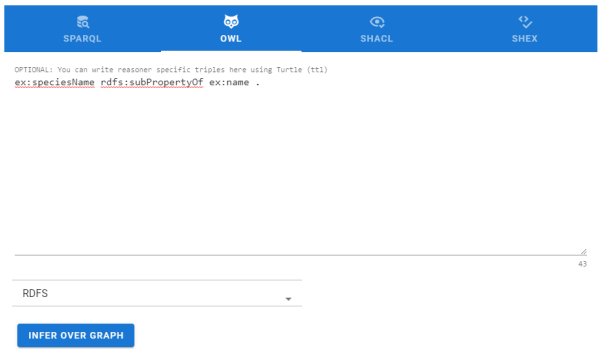

Figura 3.5

Figura 3.6: (a) campo de SPARQL para una actualización de la TDB de tipo *INPUT* (ingreso o entrada de datos). (b) mensaje de resultado de la actualización expresada en SPARQL para la TDB.(c) nuevo estado de la TDB luego de la operación SPARQL Update. (d) vista del nuevo estado de la TDB luego de la operación SPARQL Update.

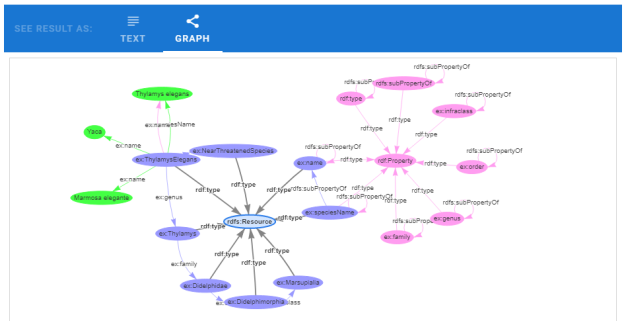
### 3.2.4. Realizar razonamiento en RDFS/OWL de un grafo

Para este caso de uso se considera que el estudiante ya ha descrito un grafo RDF correctamente utilizando el sintaxis *Turtle* (caso de uso 3.2.1); luego de lo cual debe agregar triples para que el razonador entienda que en el triple `ex:ThylamysElegans ex:speciesName "Thylamys elegans"@la .`, el nombre de la especie (también denominado nombre científico) expresado como `es:speciesName` es también un "nombre" (`ex:name`), de la misma forma que "`Yaca`" en español de Chile.

Entonces el estudiante en la zona de herramientas presiona la pestaña "*OWL*" (figura 3.7a), encontrándose con un área de texto donde puede agregar instrucciones en sintaxis *Turtle*, en la que ingresa el siguiente triple `ex:speciesName rdfs:subPropertyOf ex:name .`, selecciona en el menú desplegable la opción "*RDFS*" y presiona el botón "*INFER OVER GRAPH*". RDF Playground muestra el resultado del razonamiento en la sección de resultados, tanto en texto *Turtle* como en forma de visualización, en la cual los triples agregados por el razonador son destacados con color fucsia (figura 3.7b).




(a)

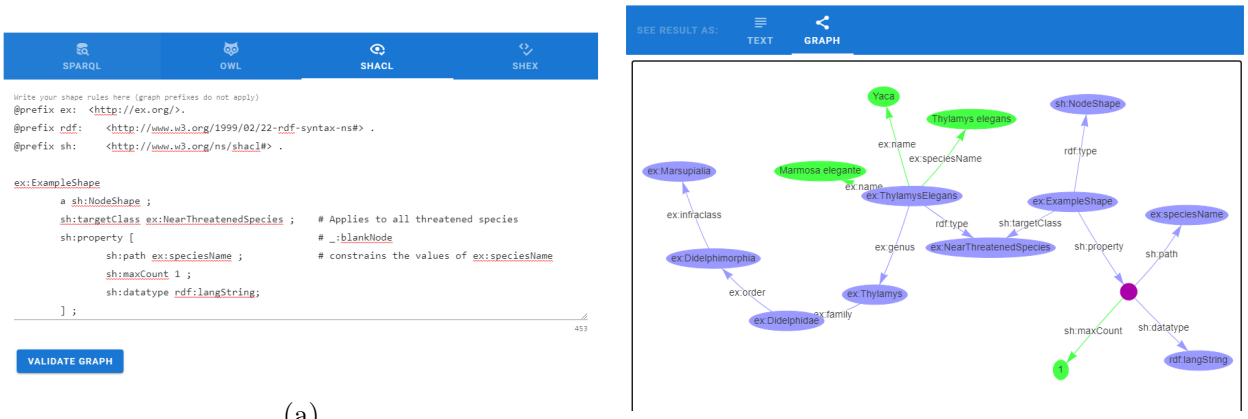



(b)

Figura 3.7: (a) pestaña de la herramienta de razonamiento. (b) visualización del resultado del razonamiento, con los nuevos arcos y nodos de color fucsia.

### 3.2.5. Validación de grafo utilizando SHACL

Este caso también considera que se ha descrito un grafo RDF correctamente (ver caso de uso 3.2.1); para el cual el estudiante debe validar que cada `ex:NearThreatenedSpecies` posea a lo más un nombre científico (`ex:speciesName`); para ellos el estudiante va a la pestaña "*SHACL*" de la zona de herramientas (figura 3.8a), ingresa los prefijos a utilizar, y siguiendo las indicaciones del cuerpo docente crea una regla que el considera que valida esta condición, luego presiona el botón "*VALIDATE GRAPH*"; entonces RDF Playground muestra en la sección de resultados el mensaje "*The data graph conforms with the shape graph!*", y en la visualización muestra la unión de las reglas con el grafo descrito (figura 3.8b).

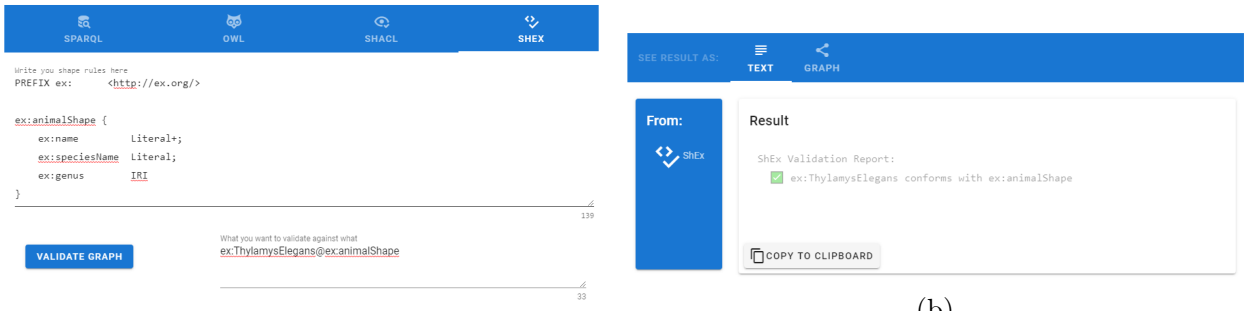



(a) (b)

Figura 3.8: (a) pestaña de la herramienta de validación de SHACL. (b) visualización del grafo descrito junto a las reglas agregadas.

### 3.2.6. Validación del grafo utilizando ShEx

El siguiente caso de uso considera que se ha descrito un grafo RDF correctamente (ver caso de uso 3.2.1); el estudiante debe validar que una especie tenga uno o más nombres como literal; un nombre científico como un literal y un genero como IRI; para completar la actividad el estudiante va a la pestaña "*SHEX*" de la zona de herramientas (figura 3.9a), donde en el campo superior (correspondiente a las reglas que un grafo debe cumplir), describe las reglas de la forma que se aprecia en la figura 3.9a, luego en el campo de más abajo señala con el texto `ex:ThylamysElegans@ex:animalShape`, que va a someter a `ex:ThylamysElegans` a la validación de la forma `ex:animalShape`; después presiona el botón "*VALIDATE GRAPH*" para correr la validación. RDF Playground muestra el reporte en forma de texto en la sección de respuesta (ver figura 3.9b).




(a) (b)

Figura 3.9: (a) pestaña de la herramienta de validación de ShEx. (b) reporte entregado por el validador de ShEx.

# Capítulo 4

# Diseño de la Aplicación

Tal como ha sido mencionado RDF Playground busca facilitar el aprendizaje de los contenidos del curso "La Web de Datos", con una interfaz que permita visualizar y editar grafos RDF, además de explotar las capacidades que son parte del contenido del curso. El proceso de diseño y desarrollo se ha seguido el principio de las metodologías ágiles, añadiendo capacidades a un producto siempre funcional.

## 4.1. Arquitectura de la Solución

La estructura de RDF Playground está compuesta por un *frontend* y un *backend*, componentes independientes entre sí. Se decide dividirlo en servicios independientes para agilizar el desarrollo, facilitar escalar el sistema en el futuro, y permitir cambios rápidos en el servicio de ser necesario.

Un estudiante interactúa con el sistema a través de una página web estática, que fue desarrollada utilizando *Vue*, apoyado con un *framework* para utilizar una interfaz de usuario (UI) del estilo de *Material Design* [13]; además, el *frontend* permite visualizar el grafo RDF cuando este es recibido en el lenguaje DOT, mediante la biblioteca *VisJs* [5]; la página web se comunica con el *backend* para delegar el procesamiento de la información en RDF.

El *frontend* del sistema muestra al usuario cuatro secciones (ver figura 4.1): un lienzo con el ícono y nombre del sistema, junto a la inscripción "*Universidad de Chile*"; una sección que ocupa la mitad izquierda del espacio horizontal, donde el estudiante describe grafos RDF en sintaxis *Turtle*, pudiendo verificar la sintaxis de dicha descripción y visualizarla tanto como texto o como grafo; una sección de herramientas que ocupa la mitad derecha del espacio descrito anteriormente, con la que el usuario puede acceder a varias herramientas vistas durante el curso; y una sección inferior de resultados, en la cual se muestra en texto o visualización el resultado de operar un grafo con las distintas herramientas de la sección anteriormente descrita. Esta sección muestra al estudiante qué herramienta generó el resultado, además de permitir copiar su contenido al portapapeles para que sea utilizado como este estime conveniente.

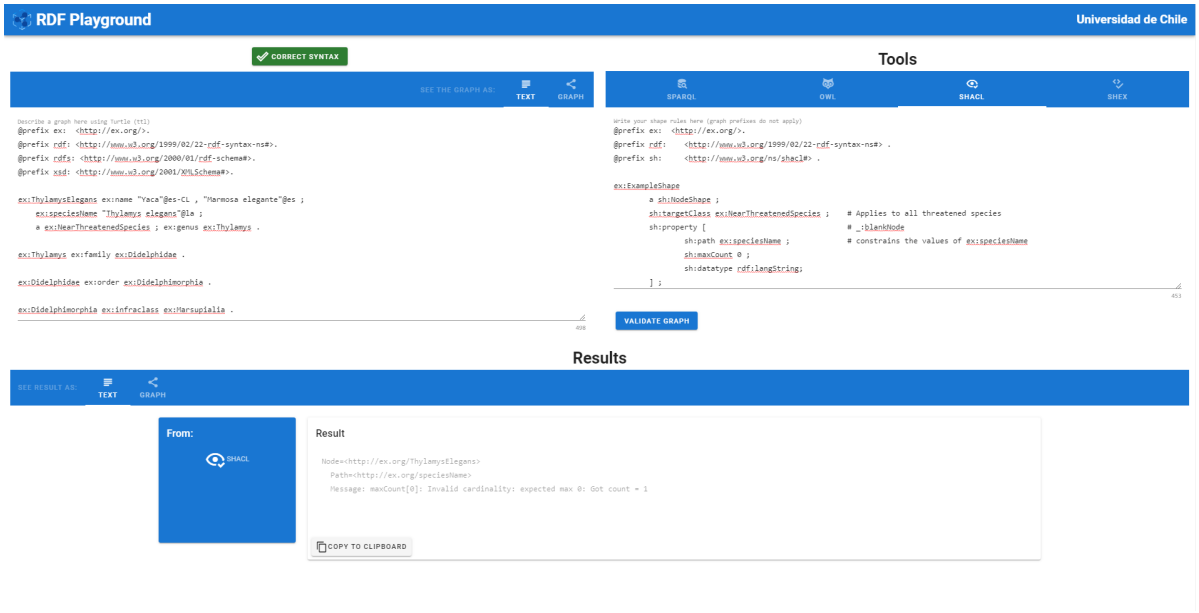



Figura 4.1: Captura de pantalla del *frontend* de RDF Playground

El *backend* es una *API REST* desarrollada en el lenguaje Kotlin (la cual opera sobre la máquina virtual de Java, y a diferencia del lenguaje Scala puede trabajar en conjunto con bibliotecas y código escrito en el lenguaje Java), la cual implementa el grueso de las funcionalidades y la totalidad del procesamiento del sistema RDF Playground. Este maneja la comunicación con el *frontend* a través del protocolo HTTP utilizando la plataforma *Spring Boot* implementada en Java. A continuación se describen los principales componentes y sus funcionalidades.

La biblioteca *Apache Jena* se utiliza como la base para el procesamiento de grafos en RDF, además de proveer razonamiento con el lenguaje SHACL sobre grafos RDF, una base de triples de datos y la capacidad de realizar operaciones SPARQL sobre grafos dados o sobre la base de triples de datos. La biblioteca es modular y extensible, separando las distintas capacidades en distintas sub-bibliotecas; en particular para este trabajo de título se ha extendido la API de RDF, que maneja grafos en varias sintaxis definidas por la W3C, para agregar la capacidad de transformar un modelo al lenguaje DOT, con el objetivo de que las partes que interactúan con el backend puedan utilizar herramientas gráficas para generar visualizaciones a partir del modelo expresado en DOT.

El sistema utiliza una versión modificada de la biblioteca *OWL-RL* para este trabajo de título por el autor; la biblioteca fue creada por la organización de *GitHub* RDFLib en el lenguaje Python. *OWL-RL* provee razonamiento RDFS y OWL 2 RL para grafos descritos en de RDF, el cual utiliza el sistema para comunicarse con esta biblioteca mediante un *script* que esta pone a disposición y que corre este sistema desde una línea de comandos. La modificación realizada elimina el soporte de la biblioteca para los triples generalizados (definidos por la W3C como triples con sujeto, objeto y predicado, donde cada uno de estos puede ser un IRI, nodo blanco o literal [8, sección 7]), los cuales si bien son soportados por RDFLib y sus bibliotecas, no son parte del estándar establecido por la W3C.

Para proveer al sistema con la capacidad de realizar validaciones en el lenguaje ShEx se utiliza la biblioteca *ShExJava*, la cual se integra con el resto del *backend* mediante la transformación de los grafos de la API de RDF de *Apache Jena* a las clases manejadas por *ShExJava* y *vice versa*.

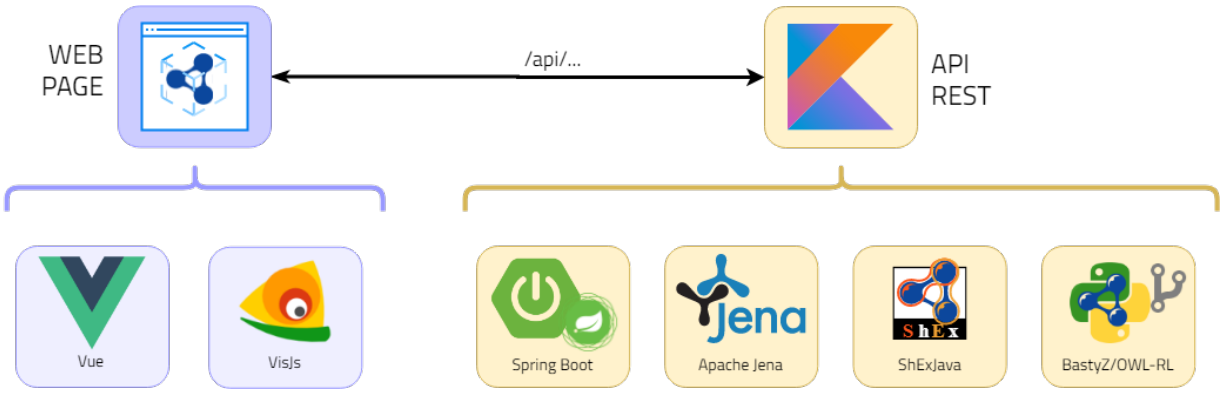



Figura 4.2: Arquitectura del sistema RDF Playground

## 4.2. Lógica de la Aplicación

Cada una de las herramientas mencionadas en la sección 3.2 interactúa con el lado del servidor mediante el uso de una interfaz de programación de aplicaciones (API, acorde a sus iniciales en inglés); esta se ha diseñado de forma que el servidor no guarde estados, preferencias ni conexiones, además de incluir algunas opciones que aún no se utilizan en el *frontend* pensadas para posibles extensiones del sistema.

Todas las consultas se hacen mediante peticiones HTTP utilizando el formato JSON y las respuestas son texto o varios textos en formato JSON, incluyendo en el encabezado de esta un código, mensaje de estado de acuerdo al código y tipo de contenido (definidos según la asignación realizada por IANA [10]) de acuerdo a lo establecido en la especificación del protocolo HTTP [18, p. 57].

Los métodos disponibles en la API son agrupados en controladores según las capacidades de RDF que explotan, existiendo cuatro: controlador de modelos (`/api/model/`); controlador de base de triples de datos (`/api/tdb/`), controlador del razonador (`/api/owl/`) y controlador de formas (`/api/shape/`); estos controladores sirven para agrupar las consultas y hacer más ordenados los llamados al *backend*; en cada método se puede recibir descripciones de grafos RDF en las distintas sintaxis soportadas por Jena (por defecto el sistema espera una descripción en sintaxis *Turtle*), respondiendo en sintaxis *Turtle* (con excepción de las consultas SPARQL, las que responden en los formatos que provee *Apache Jena* para ello, incluida la extensión a DOT). A continuación se presentan los métodos principales de la interfaz de programación de aplicaciones, ordenados según cada uno de los controladores mencionados.

**Controlador de Modelos**

1. **Validación de sintaxis**
   Petición HTTP: `POST` */api/model/syntax_check*
   **Parámetros de la consulta:**

| Parámetro | Tipo | Descripción |
|---|---|---|
| data | Texto | Grafo a validar |
| data_lang | Texto | Lenguaje de descripción del grafo |

   **Parámetros de la respuesta:**
   **Tipo de contenido:** *application/json.*

| Respuesta | Tipo | Descripción |
|---|---|---|
| syntax_error | Texto | Primer error encontrado en la sintaxis |
| data_dot | Texto | Descripción del grafo en DOT (de no existir errores) |

2. **Conversión a DOT**
   Petición HTTP: `POST` */api/model/dot*
   **Parámetros de la consulta:**

| Parámetro | Tipo | Descripción |
|---|---|---|
| data | Texto | Grafo a ser convertido a DOT |
| data_lang | Texto | Lenguaje de descripción del grafo |

   **Parámetros de la respuesta**
   **Tipo de contenido:** *text/vnd.graphviz.*

| Respuesta | Descripción |
|---|---|
| data_dot | Descripción del grafo en DOT |

**Controlador de base de triples de datos**

1. **Verificación de disponibilidad de la base de triples de datos**
   Petición HTTP: `POST` */api/tdb/status*
   **Parámetros de la respuesta:**
   **Tipo de contenido:** *text/plain*

| Respuesta | Descripción |
|---|---|
| tdb_status | Descripción de estado acorde a estado HTTP |

2. **Consulta a la base de triples de datos**

   Petición HTTP: `POST` */api/tdb/query_tdb*

   **Parámetros de la consulta:**

| Parámetro | Tipo | Descripción |
|---|---|---|
| query | Texto | Consulta SPARQL a la base de triples de datos |
| type | Texto | Tipo MIME de la respuesta solicitada |

   **Parámetros de la respuesta:**

   **Tipo de contenido:** *text/plain* en caso de error o el tipo solicitado en la consulta.

| Respuesta | Tipo | Descripción |
|---|---|---|
| query_response | Texto | Respuesta a la consulta SPARQL |

3. **Consulta SPARQL con un grafo dado**

   Petición HTTP: `POST` */api/tdb/query_model*

   **Parámetros de la consulta:**

| Parámetro | Tipo | Descripción |
|---|---|---|
| data | Texto | Grafo sobre el cual se realiza la consulta |
| data_lang | Texto | Lenguaje en que se describe el Grafo |
| query | Texto | Consulta SPARQL a la base de triples de datos |

   **Parámetros de la respuesta:**

   **Tipo de contenido:** *text/plain* en caso de error o el tipo solicitado en la consulta.

| Respuesta | Tipo | Descripción |
|---|---|---|
| query_response | Texto | Respuesta a la consulta SPARQL |

4. **Obtener contenido de la base de triples de datos**

   Petición HTTP: `GET` */api/tdb/get_model*

   **Parámetros de la respuesta:**

   **Tipo de contenido:** *application/json.*

| Parámetro | Tipo | Descripción |
|---|---|---|
| data | Texto | Contenido de la TDB en formato *Turtle* |
| dot | Texto | Contenido de la TDB en formato DOT |

**Controlador del Razonador**

1. **Razonar sobre un modelo**

   Petición HTTP: `POST` */api/owl/reason*

   **Parámetros de la consulta:**

| Parámetro | Tipo | Descripción |
|---|---|---|
| data | Texto | Grafo sobre el cual se razona |
| data_lang | Texto | Lenguaje en que se describe el Grafo |
| profile | Texto | Esquema a usar (RDFS u OWL [OWL 2 RL]) |

   **Parámetros de la respuesta:**

   **Tipo de contenido:** *application/json* o *text/plain* en caso de error de servidor.

| Parámetro | Tipo | Descripción |
|---|---|---|
| data | Texto | Grafo con resultados del razonador |
| error | Texto | Descripción de error en el grafo descrito |
| data_dot | Texto | Grafo con resultados del razonador en DOT |

2. **Razonar con tiempo límite**

   Petición HTTP: `POST` */api/owl/reason_timeout*

   **Parámetros de la consulta:**

| Parámetro | Tipo | Descripción |
|---|---|---|
| data | Texto | Grafo sobre el cual se razona |
| data_lang | Texto | Lenguaje en que se describe el Grafo |
| profile | Texto | Esquema a usar (RDFS u OWL [OWL 2 RL]) |
| timeout | Entero | Tiempo límite de razonamiento |

   **Parámetros de la respuesta:**

   **Tipo de contenido:** *application/json* o *text/plain* en caso de error de servidor o fin del tiempo determinado.

| Parámetro | Tipo | Descripción |
|---|---|---|
| data | Texto | Grafo con resultados del razonador |
| error | Texto | Descripción de error en el grafo descrito |
| data_dot | Texto | Grafo con resultados del razonador en DOT |

**Controlador de Formas**

1. **Validación de ShEx**

   Petición HTTP: `POST` */api/shape/shex_isvalid*

   **Parámetros de la consulta:**

| Parámetro | Tipo | Descripción |
|---|---|---|
| data | Texto | Grafo a validar |
| data_lang | Texto | Lenguaje del Grafo a validar |
| shape | Texto | Grafo de formas |
| shape_map | Texto | Lista de entidades y formas a cumplir |

   **Parámetros de la respuesta:**

   **Tipo de contenido:** *text/plain.*

| Respuesta | Tipo | Descripción |
|---|---|---|
| validation_report | Texto | Reporte de validación ShEx |

2. **Validación de SHACL**

   Petición HTTP: `POST` */api/shape/shacl_isvalid*

   **Parámetros de la consulta:**

| Parámetro | Tipo | Descripción |
|---|---|---|
| data | Texto | Grafo a validar |
| data_lang | Texto | Lenguaje del Grafo a validar |
| shape | Texto | Grafo de formas |
| shape_lang | Texto | Lenguaje del grafo de formas |

   **Parámetros de la respuesta:**

   **Tipo de contenido:** *application/json* o *text/plain* en caso de error al realizar el reporte.

| Parámetro | Tipo | Descripción |
|---|---|---|
| report | Texto | Reporte de validación (vacío en caso de éxito) |
| fusion_dot | Texto | Grafo con *data* y *shape* en lenguaje DOT |

La API descrita provee a RDF Playground con el procesamiento necesario, cubriendo la totalidad de las capacidades vistas en el curso, de forma tal que no guarde estados ni mantenga conexiones, siendo extensible y dando flexibilidad al sistema, ya que se puede correr más de una instancia de la API, o esta misma puede responder a más de un *frontend* configurando políticas de Intercambio de Recursos de Origen Cruzado (CORS) en *Spring Boot*.

# Capítulo 5

# Evaluación de la Solución

En esta sección se presenta la evaluación de una de las últimas versiones del sistema, la cual no incluye la diferenciación en la visualización de los grafos RDF presente en la historia de usuario 3.2.4 (figura 3.7b); además se añade en la sección 5.4 una colección de opiniones de los ayudantes y auxiliares del cuerpo docente, que al haber estado en ediciones anteriores del curso conocen las herramientas utilizadas en los laboratorios de estas.

Esta versión es presentada a los estudiantes del curso "La Web de Datos" en la edición de primavera de 2020 (el curso se realiza de manera anual, en el semestre de primavera), realizado de manera completamente remota, a diferencia de las versiones anteriores del curso, entre los meses de agosto y diciembre de 2020. A continuación se describe el proceso que siguió esta evaluación, y los resultados obtenidos.

## 5.1. Metodología de evaluación

En este proceso se le presentó el sistema RDF Playground a los usuarios durante el segundo y tercer laboratorio del curso; los laboratorios fueron realizados de manera remota utilizando la plataforma *Discord* (plataforma de conversación por texto, voz y vídeo similar a *Slack*, orientada a videojuegos), donde los participantes trabajaron en grupos de hasta tres personas en comunicación con el cuerpo docente; los usuarios interactuaron con el sistema desde sus navegadores, encontrándose en comunicación constante con el cuerpo docente; además se realizó la misma evaluación con dos usuarios expertos, uno de ellos auxiliar del mismo curso.

En las diferentes secciones los usuarios interactuaron con el sistema tal cual como este se encontraba, sin haber utilizado ni tener conocimiento de los sistemas utilizados en versiones anteriores del curso; por los laboratorios en que se realizaron las evaluaciones los estudiantes no utilizaron todas las capacidades del sistema, si no que en la primera utilizaron solamente la capacidad de describir grafos RDF, revisando la sintaxis de estos y utilizando visualización del grafo descrito, siendo la primera interacción con las herramientas de RDF; mientras que en la segunda utilizaron las capacidades de la primera evaluación en conjunto con la pestaña "*OWL*", y las capacidades de razonamiento sobre el grafo.

Al terminar cada una de estas sesiones a los participantes se les pidió contestar un cuestionario con diez preguntas, utilizando una escala "Likert" de 5 puntos para especificar las respuestas. La totalidad de las preguntas son parte de la encuesta de usabilidad de sistema (SUS por su sigla en inglés) [4], solicitando además comentarios generales para mejorar el sistema.

## 5.2. Descripción de los participantes

Diez personas participaron en cada una de las encuestas, incluyendo a un experto y al auxiliar entre estos participantes, cuyas respuestas fueron hechas mediante el mismo medio que los estudiantes. En adelante se hace la diferencia entre cada una de las encuestas para el análisis. A cada participante se le preguntó cuanto tiempo llevaba programando en general, y sobre su interés en la web semántica.

### 5.2.1. Encuesta de primera impresión de sistema (laboratorio 2)

Los participantes llevan en su mayoría menos de seis años programando, existiendo solamente tres que llevan más que ese tiempo; de los siete usuarios que llevan menos de seis años programando la mayoría llevan entre dos y tres años haciéndolo (figura 5.1). Al preguntarles por su interés en la web semántica, la mayoría (7) declara que está recién conociendo la materia, lo cual corresponde con el carácter de electivo de este curso (figura 5.2).

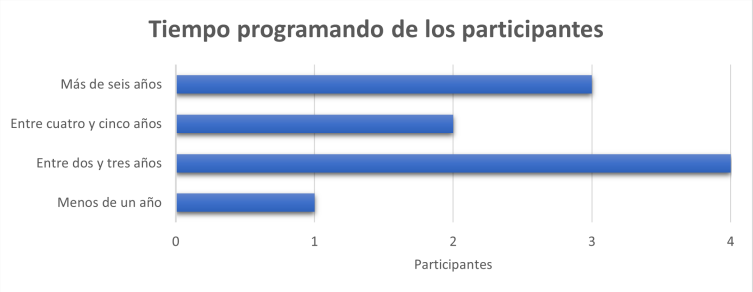



Figura 5.1: Caracterización de las respuestas según tiempo de programación

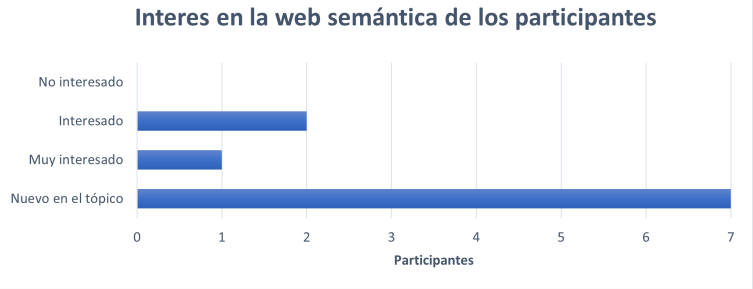



Figura 5.2: Caracterización de las respuestas según interés en el tópico

### 5.2.2. Encuesta de la primera experiencia con OWL (laboratorio 3)

Para este laboratorio los participantes varían un poco en su composición, en general el tiempo que llevan programando es similar; pero a diferencia del laboratorio anterior muchas más personas se declaran nuevas en el tópico 5.4, de lo cual podemos inducir que la cantidad de expertos en esta encuesta es mucho mayor que la anterior.

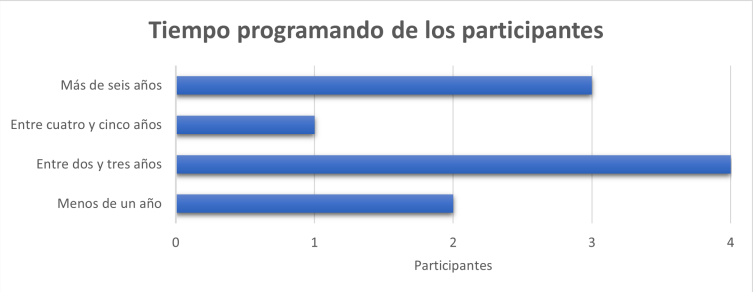



Figura 5.3: Caracterización de las respuestas según tiempo de programación


Figura 5.4: Caracterización de las respuestas según interés en el tópico

## 5.3. Evaluación de usabilidad del sistema

En esta sección se analizan por separado los resultados de las dos encuestas de usabilidad de sistemas (SUS) [4], donde se les pide a los participantes que respondan su opinión respecto de 10 sentencias con cinco posibles respuestas, variando desde "Muy en desacuerdo" (representado con puntaje 1) a "Muy de acuerdo" (puntaje 5); las sentencias se van intercalando entre "positivas" y "negativas", por lo que se espera de las primeras un puntaje alto, y de las últimas un puntaje bajo; el puntaje final (ver la figura 5.5) es calculado asignando puntaje a cada pregunta); luego normalizando el valor de las preguntas negativas a la misma escala de las positivas con el objeto de tratar todas las sentencias de la misma manera, se suman los resultados y se multiplican por el valor 2,5. A continuación se muestran los resultados de esta encuesta para los dos laboratorios evaluados.

$$2{,}5 \times \left( \sum_{\mathrm{i}=1}^{5} p_{2\mathrm{i}-1} - 1 + \sum_{\mathrm{i}=1}^{5} 5 - p_{2\mathrm{i}} \right)$$

Figura 5.5: Calculo de puntaje final SUS, donde $p_{\mathrm{i}}$ corresponde a una pregunta

### 5.3.1. Primera encuesta (laboratorio 2)

En este laboratorio se le entregó a los participantes un ejemplo de grafo descrito en sintaxis *Turtle*; se les pide copiarlo en RDF Playground, validar su sintaxis y visualizarlo para familiarizarlos con el sistema; luego se les pide que lean un párrafo escrito en prosa y se les pide que describan la información allí presente como un grafo RDF, entregando esta descripción para finalizar la actividad.

La tabla muestra las preguntas realizadas a los usuarios con el fin de determinar la usabilidad del sistema, según su percepción. Si bien la cantidad de respuestas no alcanzan para creer que los resultados se ajustarán totalmente a la frecuencia de un usuario cualquiera (al ser solo 11 respuestas), los participantes evaluaron el sistema con una puntuación de 67,5 (tabla 5.1) que lo deja muy cerca de ser considerada usable, pero sin superar el límite en que a un sistema normalmente se le considera usable ($>$ 68 pts.).

| | Total | |
|---|---|---|
| **Sentencia a evaluar** | **P** | **DE** |
| Me gustaría usar frecuentemente este sistema | 4 | 0,78 |
| Encontré el sistema innecesariamente complejo | 1,64 | 0,5 |
| Opino que el sistema es fácil de usar | 4,55 | 0,52 |
| Creo que necesitaría ayuda para poder usar este sistema | 2,18 | 0,6 |
| Considero que la distribución de elementos en el sistema es correcta | 3,91 | 0,83 |
| Pienso que había mucha inconsistencia en este sistema | 1,63 | 0,67 |
| Creo que es fácil aprender a utilizar este sistema | 4,18 | 0,87 |
| Creo que el sistema era muy complicado de usar | 1,73 | 0,65 |
| Me sentí muy seguro al usar el sistema | 4,09 | 0,54 |
| El sistema me dificultó el aprendizaje | 1,55 | 0,52 |
| ***Puntaje SUS*** | 67,5 | 7,24 |

Tabla 5.1: Respuestas de encuesta SUS de primera impresión del sistema (laboratorio 2). **P:** promedio **DE:** desviación estándar.

Los participantes no creen del todo que la distribución de los elementos sea correcta, lo que se relaciona bastante con las sugerencias entregadas de dar más espacio a la visualización del grafo. De las cosas positivas más destacables tenemos que a los participantes les gustaría usar frecuentemente el sistema, lo encuentran fácil de usar y se sienten seguros al usar el sistema; una cosa importante es como los estudiantes evalúan si el sistema les dificultó el aprendizaje, donde la respuesta marca un claro desacuerdo con la afirmación.

De los comentarios adicionales entregados por los participantes todos se centran en las siguientes mejoras: agregar número de línea en los campos de texto de la interfaz, agrandar el

tamaño de la visualización del grafo, permitir "minimizar y ocultar" la zona de herramientas, y se hace una mención a la posibilidad de descargar el grafo descrito como un archivo. Varios de los comentarios están enfocados a la cantidad de espacio que utiliza la visualización del grafo, ofreciendo incluso ideas de solución a este problema como sugerencias, lo cual marca una correlación clara con el puntaje relativamente bajo acerca de la distribución de los elementos de la interfaz.

Con esto concluimos para esta encuesta que el sistema en general cumple su objetivo de apoyar el aprendizaje, pero aún necesita ajustes en pos de ser más intuitivo y hacer a los estudiantes menos dependientes de la ayuda para usar el sistema, especialmente en lo referente al tamaño de las visualizaciones, y la presencia de números de línea en los textos para facilitar la navegación por estos.

### 5.3.2. Segunda encuesta (laboratorio 3)

En este laboratorio se les entrega a los participantes una descripción de grafo RDF con personajes mitológicos y legendarios, que incluye sus nombres, el significado de estos y su alineamiento basado en el alineamiento del juego de rol "Dungeons & Dragons" [17], se les presenta la pestaña "OWL" y se les pide crear e introducir nuevos triples utilizando elementos del esquema RDFS (ej. `ex:Deity rdfs:subClassOf ex:God .`) con el fin de que el razonador de RDFS concluya algunos triples específicos dados en el enunciado (por ejemplo, se busca que el razonador concluya `ex:Vishnu a ex:God .` basado en el triple del ejemplo anterior y la presencia del siguiente triple en el grafo original `ex:Vishnu a ex:Deity .`).

En esta encuesta las respuestas tampoco alcanzan para asegurar que se ajustarán a un usuario promedio, siendo solo diez respuestas, por lo que ponemos especial atención a la desviación estándar para realizar el análisis. Los estudiantes en esta ocasión, utilizando el sistema para describir un grafo y razonar sobre este con la herramienta de la pestaña "*OWL*" dieron al sistema un puntaje ligeramente más bajo (64,5 pts., tabla 5.2) respecto al de la encuesta anterior, nuevamente cerca de hacer el sistema usable pero no sobre el puntaje de 68 puntos requerido.

De las respuestas de los participantes notamos que siguen en general la tendencia marcada por la primera encuesta, con las variaciones más notables en que los participantes están más en desacuerdo con que necesitarían ayuda para usar el sistema, posiblemente dado a su experiencia anterior con este; baja la percepción de la correctitud de la distribución de elementos en el sistema y los participantes se sintieron mucho más seguros al utilizar el sistema. Hay que tener en cuenta la variación en la desviación estándar de las sentencias "Creo que necesitaría ayuda para poder usar este sistema" que aumenta de 0.6 a 1.25 y "Me sentí muy seguro al usar el sistema " que pasa de 0.54 a 1.17, en ambos casos la evaluación de la sentencia es más moderada, lo que se puede interpretar como un encuentro con más elementos poco intuitivos, ya que por ejemplo ha empeorado la puntuación en la sentencia sobre la complejidad del sistema la sentencia sobre la inconsistencia de este. Se toma en cuenta que la actividad a realizar en este laboratorio es considerablemente más compleja que la del laboratorio anterior, y los participantes utilizan en esta ocasión los paneles de edición de grafos, herramientas y resultados a la vez, lo que también puede ayudar a explicar el

| Sentencia a evaluar | Total P | Total DE |
|---|---|---|
| Me gustaría usar frecuentemente este sistema | 3,8 | 0,42 |
| Encontré el sistema innecesariamente complejo | 1,7 | 0,48 |
| Opino que el sistema es fácil de usar | 4,6 | 0,51 |
| Creo que necesitaría ayuda para poder usar este sistema | 2,7 | 1,25 |
| Considero que la distribución de elementos en el sistema es correcta | 4,1 | 0,74 |
| Pienso que había mucha inconsistencia en este sistema | 1,8 | 0,92 |
| Creo que es fácil aprender a utilizar este sistema | 4,3 | 0,95 |
| Creo que el sistema era muy complicado de usar | 1,7 | 0,67 |
| Me sentí muy seguro al usar el sistema | 3,4 | 1,17 |
| El sistema me dificultó el aprendizaje | 1,5 | 0,52 |
| ***Puntaje SUS*** | 64,5 | 9,34 |

Tabla 5.2: Respuestas de encuesta SUS encuesta luego de usar OWL sin diferenciación de nodos generados en la visualización (laboratorio 3). **P:** promedio **DE:** desviación estándar.

menor puntaje, especialmente respecto a la sentencia acerca de la complejidad del sistema.

De los comentarios sacamos opiniones como que existen notificaciones de error que no se cierran al no existir más el error, a menos que sean cerradas manualmente; se repite la idea de poder copiar o descargar la descripción del grafo creada, esta vez relacionada a otro comentario que dice que el comando de copiar pegar no funciona siempre, y un comentario en el que el participante señala que hay funciones del sistema que aún no entiende, en su opinión por falta de conocimiento, por lo que le gustaría que se añadieran indicaciones de que hace cada botón.

De los resultados de la encuesta notamos que aparecen elementos que se repiten relacionados con lo esperado por los participantes del sistema, dada su experiencia con las nuevas herramienta del sistema, como que los errores desaparezcan automáticamente cada vez que vuelven a correr una simulación, o las funciones de copiar y pegar que son nativas de cada navegador. Se nota que el sistema puede ser más explicativo, y quizás se pueda mejorar para que sea más fácil descubrirlo sin la necesidad de apoyo de el cuerpo docente del curso.

## 5.4. Opinión de ayudantes y auxiliares del curso

Los ayudantes y auxiliares de la edición primavera 2020 del curso "La Web de Datos", a diferencia de los estudiantes de esta edición del curso, han tenido la oportunidad de experimentar las herramientas anteriores del curso como estudiantes y en sus cargos dentro del cuerpo docente de otras ediciones del curso, al mismo tiempo de poder ver y explorar RDF Playground durante la edición actual del curso; lo que les da una perspectiva bastante amplia sobre la utilidad de RDF Playground en su principal objetivo de facilitar el aprendizaje de las contenidos del curso (ver sección 1.2.1). Dado que son en total cuatro personas, se les pidió evaluar cualitativamente la herramienta respecto a las otras herramientas utilizadas en versiones anteriores del curso, a través de las siguientes preguntas:

1. ¿Fuiste auxiliar o ayudante en versiones anteriores del curso?
2. ¿En que año realizaste el curso como estudiante?
3. ¿Crees que la herramienta es mejor que las herramientas usadas en versiones anteriores del curso? ¿Por qué?
4. ¿Cuáles son las ventajas principales del nuevo sistema?
5. ¿Cuáles son las desventajas principales del nuevo sistema?

En las respuestas nos encontramos con que en general las personas que respondieron la encuesta han sido auxiliares en más de una ocasión; se considera una ventaja comparativa importante que RDF Playground tenga las capacidades integradas dentro de la herramienta, disminuyendo en su opinión la «confusión de los estudiantes» que tenían en versiones anteriores del curso para lograr las actividades solicitadas; se señala como desventaja principal la existencia *triples axiomáticos* al utilizar los razonadores de RDFS y OWL, especialmente al visualizar estos; estos triples son una cantidad considerable, del orden de las decenas para RDFS y fácilmente superando las centenas en OWL, y no aportan conocimiento relevante en los laboratorios; dada la gran cantidad de estos, la visualización del resultado se ve particularmente afectada, ya que estos triples axiomáticos ocupan bastante espacio respecto a los grafos que comúnmente se trabajan en los laboratorios. En el caso particular de RDFS, además crean *clusters* de datos que no son relevantes para los laboratorios.

## 5.5. Estabilidad del sistema y servicio

Al momento de redactar este documento el RDF Playground lleva cinco semanas corriendo, lo cual corresponde a cuatro laboratorios dada la existencia de una semana de receso universitario a la segunda semana; el sistema no ha presentado problemas de intermitencia en el servicio, ni comportamientos no deseados; soportando hasta 40 conexiones simultaneas en la máquina virtual del curso, la cual posee cuatro gigabytes de memoria RAM.

Es de nuestra consideración que el sistema ha cumplido ampliamente con la estabilidad esperada, incluso considerando que para la implementación del sistema la cantidad de RAM fue elegida por el uso que compilar una versión de producción del *frontend* hace de esta (cantidad que en general se acerca a los tres gigabytes), sin realizar un análisis previo de la utilización de memoria de este tipo por parte de RDF Playground. El sistema en el tiempo que ha estado corriendo solo ha sido detenido para introducir las mejoras y ajustes que se le ha hecho al *backend*, que en general no requieren detener el sistema por más de un minuto; para el *frontend*, en cambio, se ha logrado introducir ajustes y mejoras sin interrumpir el servicio, consecuencia natural de que las versiones de producción generadas son páginas estáticas.

La ausencia de interrupciones en el servicio es particularmente relevante dado que la entrega del trabajo de los laboratorios se realiza en el mismo formato que una tarea: los estudiantes utilizan el horario de laboratorio para trabajar y realizar consultas al cuerpo docente en tiempo real, pero pueden entregar sus resultados en el plazo de una semana; luego de la cual pueden hacer entrega de esta pero se considera atrasada. Este formato implica que los estudiantes pueden avanzar en los laboratorios en cualquier momento, por lo que tener interrupciones en cualquier momento podría limitar el avance de estos.

# Capítulo 6

# Conclusión

RDF Playground busca facilitar el aprendizaje los contenidos del curso "La Web de Datos", mediante una aplicación web que permite poner en práctica todos los tópicos del curso; con una interfaz que permite trabajar los datos en forma de texto, y visualizar tanto los datos como el resultado de la explotación de las capacidades de RDF que expresen dichos resultados en sintaxis RDF.

El sistema desarrollado en este trabajo de título logra permitir a los estudiantes utilizar la sintaxis base del curso para explotar todas las capacidades RDF vistas en "La Web de Datos". Funcionando de manera estable e ininterrumpida como la única herramienta utilizada en una edición del curso, con alrededor de 50 estudiantes conectados a la vez.

La interfaz del sistema permite observar los datos en sintaxis RDF como texto o en una visualización, pudiendo observarlos de la segunda forma independiente de que sean estos datos ingresados por el estudiante o entregados por alguna de las herramientas dispuestas en la interfaz, habilitando estas incluso para SPARQL, capacidad que no es parte de los objetivos establecidos al comienzo de este trabajo respecto a visualizaciones; además, se logra que las visualizaciones del sistema sean interactivas, lo cual no estaba considerado inicialmente, permitiendo a los estudiantes mover un nodo en particular, incidiendo en la posición del resto, y aplicar el efecto de acercar o alejar la imagen (conocido como zum, del inglés *zoom* utilizado coloquialmente). Las visualizaciones son valoradas positivamente tanto por estudiantes como por auxiliares y ayudantes, validando su aporte al aprendizaje de los estudiantes.

En la etapa final del desarrollo de RDF Playground, se coloca a disposición de los estudiantes de la versión primavera 2020 de "La Web de Datos" el sistema, realizando dos encuestas de usabilidad a los estudiantes y una encuesta de opinión a los ayudantes y auxiliares del curso; los resultados de las encuestas realizadas muestran una valoración positiva del sistema, sin embargo, ambos grupos dejan ver que la usabilidad del sistema posee muchas oportunidades de mejora, especialmente respecto a la facilidad para que un usuario nuevo explore las capacidades ofrecidas sin una persona que haga de guía, o la expresividad de los errores y campos de texto dentro de la interfaz del sistema.

# Capítulo 7

# Trabajo Futuro

Actualmente, RDF Playground utiliza en el *backend* varias herramientas para proveer de las capacidades necesarias a la interfaz; para ello utiliza bibliotecas que funcionan tanto dentro como fuera de la máquina virtual de Java. Es deseable eliminar estas dependencias, en pos de utilizar extensiones de *Apache Jena*, u otras bibliotecas que funcionen en la máquina virtual de Java y que se puedan utilizar en conjunto a bibliotecas de Kotlin o Java.

La biblioteca *ShExJava* tuvo dos problemas a la hora de ser implementada: no soportar el uso de prefijos en el grafo de formas, lo que fue solucionado envolviendo el llamado a la herramienta, des-haciendo el uso de prefijos antes de entregar el grafo a la biblioteca; y el que su reporte solo informa si un par `entidad@forma` cumple o no las restricciones de forma para la instancia en particular, sin entregar motivos de cuál o cuáles condiciones no están satisfechas en el caso de no pasar la validación. Es deseable reemplazar o desarrollar una herramienta para la validación en ShEx que sea más expresiva en el reporte generado, para facilitar la identificación de causales de error en la validación.

El curso "La Web de Datos" se imparte utilizando principalmente la sintaxis *Turtle*, pero se espera que el sistema pueda ser de ayuda para otros cursos sobre la misma área, es por esto que la API puede recibir grafos descritos en otros lenguajes. Queda pendiente extender la API con soporte para retornar grafos RDF en otros lenguajes a petición de un cliente, para el caso en que un curso sobre el mismo tema en otra institución o un curso más avanzado sobre la web de datos pueda aprovechar el sistema para explorar los contenidos a partir de otra sintaxis.

Existen variadas ocasiones en que un estudiante solo describe un grafo, utilizando la verificación de sintaxis y la posibilidad de visualizar este, pero sin usar herramientas para consultar, razonar o validar el mismo, al trabajar de esta forma el espacio para trabajar con el grafo, especialmente el disponible para visualizar este, se hace pequeño dado que se ocupa solo una mitad de la ventana como máximo; por lo que sería deseable ajustar o rediseñar la interfaz para permitir ocultar las pestañas de herramientas, ocupando todo el espacio disponible. En la misma línea, sería ideal poder cambiar el tamaño de la visualización de los grafos RDF, en particular la altura de estos, manualmente, a gusto de los usuarios. También podría ser útil para cuando los estudiantes deban entregar informes o realizar presentaciones

añadir la opción de exportar las visualizaciones como imagen.

Al razonar con los esquemas RDFS y OWL 2 RL se producen, junto a las inferencias esperadas, varios triples con información evidente, los cuales no aportan información relevante; estos triples son denominados *triples axiomáticos*. Es deseable dar la opción de remover estos triples al procesar los datos, antes de que el grafo resultante sea mostrado en el cliente, con el objetivo de facilitar la búsqueda de información en este, tanto en el texto como en la visualización resultante.

Queda pendiente pulir el manejo de mensajes de error, especialmente en el caso en que se realiza una operación que genera un error, este se corrige sin descartar el mensaje de error, y se vuelve a realizar la operación; en este caso el mensaje actualmente no desaparece, confundiendo a los usuarios. Este mal manejo de los mensajes de error fue expresado por los participantes de la encuesta de usabilidad.

Una mejora que fue muy pedida por los participantes y que había sido considerada tempranamente en el *frontend* del sistema es que los campos en que los estudiantes ingresan el texto posean números que marquen la línea de texto; además se considera que sería beneficioso el añadir la capacidad de resaltar la sintaxis de los textos ingresados, al menos para la sintaxis *Turtle*, pero idealmente para *Turtle*, SPARQL, ShEx e incluso las otras sintaxis soportadas por Apache Jena.

Para facilitar el uso del *frontend* en los diferentes campos de texto, en especial al trabajar con grafos más grandes, sería muy útil el añadir buscadores por campo de texto, para facilitar la búsqueda de términos particulares en los grafos descritos o en los resultados al utilizar las herramientas provistas por RDF Playground.

Algo que se le puede mejorar al *frontend*, expresado por los estudiantes, son las explicaciones de qué hace cada botón, lo cual hoy debe hacer el cuerpo docente al enseñar a los estudiantes a usar el sistema. Agregar en el diseño de la interfaz pequeños mensajes explicando la función de cada elemento facilitaría a un estudiante descubrir por si mismo las capacidades soportadas y los distintos usos que permite el sistema, haciendo al mismo más atractivo para las personas que interactúen con este, especialmente en actividades no guiadas.

Al trabajar con IRIs, dado el largo de las rutas completas normalmente se utilizan prefijos, es normal intentar repetir los prefijos en los distintos documentos RDF creados. Existen servicios como `http://prefix.cc` para facilitar la búsqueda de los prefijos más utilizados para las distintas IRI. Es deseable integrar este servicio a RDF Playground utilizando los "contextos" que el servicio pone a disposición del público.

Si bien el sistema está enfocado en el aprendizaje y en las personas que están siendo introducidas a la web semántica, el que no existan herramientas que intenten unificar la experiencia, ni herramientas que exploten esta cantidad de capacidades en RDF, abre la oportunidad a crear un proyecto paralelo (o *fork*, como es denominado comúnmente en el mundo del código abierto) que sirva a los profesionales de esta área del conocimiento, ofreciendo la posibilidad de guardar sesiones, exportar las descripciones a distintas sintaxis o incluso imágenes a partir de las visualizaciones existentes.

# Bibliografía


[1] Gunnar Aastrand. Rdflib. Github repository, August 2017. https://github.com/RDFLib/rdflib/.

[2] Dave Beckett. Redland RDF libraries, 2014. http://librdf.org/.

[3] Dan Brickley, R.V. Guha, and Brian McBride. RDF Schema 1.1. W3C Recommendation, February 2014. https://www.w3.org/TR/rdf-schema/.

[4] John Brooke. Sus: a "quick and dirty" usability scale. *Usability evaluation in industry*, page 189, 1996.

[5] VisJs Community. vis.js community edition. https://visjs.org.

[6] Aba-Sah Dadzie and Matthew Rowe. Approaches to visualising Linked Data: A survey. *Semantic Web*, 2(2):89–124, 2011.

[7] Eclipse Foundation, Inc. About. https://rdf4j.eclipse.org/about/.

[8] Jeremy J. Carroll Graham Klyne and Brian McBride. RDF 1.1 Concepts and Abstract Syntax. W3C Recommendation.

[9] Pascal Hitzler, Markus Krötzsch, Bijan Parsia, Peter F. Patel-Schneider, and Sebastian Rudolph. OWL 2 Web Ontology Language Primer (Second Edition). W3C Recommendation, December 2012. https://www.w3.org/TR/owl2-primer/.

[10] IANA. Media types. https://www.iana.org/assignments/media-types/media-types.xhtml.

[11] Institut national de recherche en sciences et technologies du numerique. ShExJava: a java implementation of ShEx, 2014. http://shexjava.lille.inria.fr.

[12] Holger Knublauch and Dimitris Kontokostas. Shapes Constraint Language (SHACL). W3C Recommendation, July 2017. https://www.w3.org/TR/shacl/.

[13] John Leiden. Why you should be using vuetify. https://vuetifyjs.com/en/introduction/why-vuetify/.

[14] Alexandre Passant y Axel Polleres Paula Gearon. SPARQL 1.1 Update. W3C Recom-

mendation, mar 2013. https://www.w3.org/TR/sparql11-update/.

[15] Eric Prud'hommeaux, Iovka Boneva, Jose Emilio Labra Gayo, and Gregg Kellogg. Shape Expressions Language 2.1. W3C Final Community Group Report, October 2019. http://shex.io/shex-semantics/.

[16] Eric Prud'hommeaux and Andy Seaborne. SPARQL Query Language for RDF. W3C Recommendation, jan 2008. http://www.w3c.org/TR/rdf-sparql-query/.

[17] Lewis Pulsipher. An Introduction to Dungeons & Dragons. *White Dwarf*, 1991.

[18] R. Fielding et. al. Hypertext Transfer Protocol – HTTP/1.1. https://www.ietf.org/rfc/rfc2616.txt.

[19] RDF Working Group. Resource Description Framework (RDF). W3C Wiki, February 2014. https://www.w3.org/RDF/.

[20] Read-Write Linked Data, MIT. Linked Data API for JavaScript. Github repository, September 2019. https://github.com/linkeddata/rdflib.js.

[21] Guus Schreiber and Yves Raimond. RDF 1.1 Primer. W3C Working Group Note, June 2014. https://www.w3.org/TR/rdf11-primer/.

[22] Andy Seaborne Steve Harris and Eric Prud'hommeaux. SPARQL 1.1 Query Language. W3C Recommendation, March 2013. https://www.w3.org/TR/sparql11-query/.

[23] The Apache Software Foundation. Getting Started With Apache Jena. http://jena.apache.org/getting_started/.

[24] Wikipedia Varios. Wikidata. https://es.wikipedia.org/wiki/Wikidata.

[25] Bernard Vatant. Openlink virtuoso. W3C Wiki, February 2012. https://www.w3.org/2001/sw/wiki/OpenLink_Virtuoso.

[26] Kevin Wilkinson, Craig Sayers, Harumi A. Kuno, Dave Reynolds, and Luping Ding. Supporting scalable, persistent Semantic Web applications. *IEEE Data Eng. Bull.*, 26(4):33–39, 2003.